GAME CHANGERS

NATIONAL YOUTH SCIENCE DAY

Facilitator Guide

TABLE OF CONTENTS

KIT MATERIALS

Below is a list of the materials included in this kit. If you want to create more, printable materials are also available on the included USB drive or online at 4-h.org/NYSD.

KIT INCLUDES:

- Facilitator Guide
- Youth Workbooks (x10)
- Scratch Coding Challenge Cards (x17)
- Game Board (x5)
- Dry Erase Markers with Eraser Caps (x5)
- Tractor Game Pieces (x5)
- Obstacle Stickers (bramble, boulder and lettuce x25 each)
- Beach Balls (x2, green and yellow)

USB DRIVE INCLUDES:

- Offline versions of Scratch Desktop (for Mac OS X & Windows)
- *Pitch Your Passion* Offline Instructions
- Printable Facilitator Guide
- Printable Scratch Coding Challenge Cards
- Printable Youth Workbook
- Printable Game Board

INTRODUCTION

Welcome to *Game Changers*, the 2019 4-H National Youth Science Day (NYSD) challenge!

If you're unfamiliar with 4-H, we're the largest youth development organization in the United States, serving more than 6 million kids each year. Our philosophy is to engage kids in hands-on learning that gives them a chance to make mistakes, learn from each other and develop important life skills like communication, resilience and leadership. 4-H takes place in classrooms, clubs, afterschools and camps across the country, and covers almost any topic you can think of, from computer science to music, animal husbandry, robotics, food security and much more. Kids, volunteers and 4-H leaders can decide to pursue the topics that interest them the most. In general, 4-H projects can be grouped into four main categories, or pillar areas, which include STEM (science, technology, engineering and math), civic engagement, healthy living and agriculture. National Youth Science Day is our signature annual initiative in STEM, and is designed to help make STEM fun and accessible to young people everywhere.

This year, we've partnered with Google and West Virginia University Extension Service to create actvities that explore the topic of computer science, or CS, in a way that's fun for kids and accessible for educators everywhere — regardless of your level of access to the internet or technology. *Game Changers* consists of three activities, each of which make connections between CS topics and a different 4-H pillar — agriculture, healthy living or civic engagement — to help make CS more approachable and to help kids understand the many ways it can be applied to the world around us.

In this guide, you'll learn everything you need to know to facilitate the three *Game Changers* activities: Pitch Your Passion, Hack Your Harvest and Program Your Playground. **You don't need prior experience with computer science or coding to bring *Game Changers* to your students.** Activities are designed to make it easy for everyone, including teen teachers, to facilitate activities. Each activity includes background and preparation details for the facilitator, discussion points, prompting questions and reflection questions. *Game Changers* is perfect for first-time and beginner coders ranging in age from 8 to 14.

ICON KEY

As you use this guide, take note of the icons. Each icon indicates the type of information that appears in a given section, including instructions you can read aloud like a script, helpful facilitator tips, tie-ins to 4-H pillar areas and important vocabulary words.

READ ALOUD

FACILITATOR TIPS

IMPORTANT VOCABULARY

PILLAR TIE-IN

FACILITATOR PREPARATION

This section provides the background needed to comfortably teach the CS topics covered in this year's NYSD challenge. Read through this section first to determine which activities you'd like to use, and brush up on CS concepts that will help make facilitating a breeze. Let's get started!

Facilitator Checklist

❑ Review the basics of computer science and computational thinking in the section below.

❑ Familiarize yourself with three careers that use computer science skills.

❑ Select the activities that best fit your group and available time, space and tech.

❑ Review the vocab, materials and full instructions of the activities you choose.

❑ Print additional worksheets from the USB drive included in the kit (optional).

❑ Source any additional materials needed for the activities, including pens and pencils.

Why Computer Science (CS)?

Learning computer science is about much more than being able to write computer programs; it teaches kids an entirely new way of thinking and solving problems. In fact, recent research suggests that kids who learn CS from resourceful teachers tend to score significantly higher than their peers on standardized exams in reading, math, science and language arts. Not only that, but computer science skills, some of the most sought after in today's job market, represent an opportunity for young people of all backgrounds to achieve upward economic mobility while fulfilling a critical workforce need. As of 2019, there are half a million open computing jobs and new ones are being created at nearly four times the rate of other jobs, while paying nearly twice as well. This represents a huge opportunity for today's young people!

And computer science isn't just about coding. The hands-on CS activities in *Game Changers* teach kids essential life skills like problem-solving, teamwork and resilience, which will help prepare them for college and career.

Career examples:

- **Health:** Biomedical engineers use CS to design life-saving medical devices like insulin pumps and wearable wellness devices like fitness and health trackers.
- **Agriculture:** Farmers use CS to make their farms more efficient, from setting up precision watering systems to programming autonomous tractors and machines.
- **Civic Engagement:** Government technologists use CS to develop tools that help register voters and set policies for national cybersecurity.

What is Computational Thinking?

The process of learning computer science teaches kids many important skills, including pattern recognition, logic, problem-solving, creativity and more. While the term “computer science” describes the entire discipline of creating computer programs and systems, **computational thinking (CT)** describes skills and approaches that allow people to solve complex problems systematically.

Besides being used to develop computer applications, CT supports problem-solving across all disciplines, including math, science and the humanities. For this reason, each of the NYSD activities in *Game Changers* emphasizes computational thinking whenever possible.

Core CT concepts include:

- **Decomposition:** Breaking big problems into smaller, more manageable problems.
- **Pattern recognition:** Observing patterns, trends and regularities in data.
- **Abstraction:** Identifying and extracting relevant information to define main ideas.
- **Algorithm design:** Creating an ordered series of instructions for solving similar problems or for doing a task.

Planning Your Game Changers Event

Game Changers is adaptable to a wide range of space, time and technology constraints. Two of the activities, marked as “unplugged,” teach computer science concepts without the need for technology or devices. The computer-based CS First activity, Pitch Your Passion, includes instructions and resources that allow it to be completed with or without internet access.

The three activities in *Game Changers* can be enjoyed individually or together in any order or combination, making it a perfect fit for classrooms or outside-school learning. Activities can also be completed from start to finish in one sitting or spread out over several days. We’ve provided examples of how you might structure your NYSD event, but please feel free to experiment and find a format that works for you.

	Pitch Your Passion	Hack Your Harvest	Program Your Playground
THE FULL CHALLENGE	*120 Minutes (Activities 1 & 2)*	*45 Minutes*	*60 Minutes*
SHORT & SWEET	*30 - 60 Minutes (Activity 1)*	*45 Minutes*	
LOW-TECH		*45 Minutes*	*60 Minutes*

ACTIVITY OVERVIEWS

This section introduces the three activities that make up the *Game Changers* challenge. It gives a brief overview of each, lists the CT concepts covered, and describes each activity's tech requirements.

Pitch Your Passion

Activity URL: **g.co/csfirst/NYSDpitch**
Facilitator prep URL: **g.co/csfirst/NYSDkit**

This computer-based activity is an opportunity for kids to animate a passion, issue or cause they care about using code. By animating interactions between characters, changing their environment and adding sounds, supporters, dialogue and more, their project will help persuade others to care about their passion as well. This activity introduces kids to computer science through CS First and Scratch, a block-based coding language developed by MIT. Pitch Your Passion also has an optional follow-up activity that encourages kids to do research about a specific cause, and use that information to persuade their audience through the creation of a Public Service Announcement.

CT CONCEPTS & PILLAR TIE-IN	TOTAL TIME REQUIRED	TECH NEEDS
Computational Thinking: • Decomposition • Algorithm design **Pillar Alignment:** • Civic Engagement	**Time required:** 30-60 min **Optional follow-up activity:** 60-75 min	• Computer or tablet • Headphones (optional) • Internet connection (optional) • Projector and screen (optional)

Hack Your Harvest

This activity introduces the concepts of automation and efficiency, and teaches the basic principles of writing instructions for computers to follow. Part 1 challenges kids to find efficient solutions to puzzles that simulate programming an automated tractor. In Part 2, kids are introduced to the traveling salesman problem, a popular computer science problem, through a similar puzzle format. In Part 3, kids design their own game boards and have classmates try to solve them.

CT CONCEPTS & PILLAR TIE-IN	TOTAL TIME REQUIRED	TECH NEEDS
Computational Thinking: • Pattern recognition • Algorithm design **Pillar Alignment:** • Agriculture	**Time required: 45 min** • Part 1: 15 min • Part 2: 15 min • Part 3: 15 min	None

Program Your Playground

This activity introduces the concepts of pattern recognition and abstraction. In Part 1, kids create their own modified version of tag as they learn the concept of conditionals. They then take turns playing their new games. In Part 2, kids use pattern recognition to find the similarities in various versions of tag and work in groups to create an entirely new kind of tag. Finally, in Part 3, kids abstract concepts from several examples, then use what they learned in the previous activities to create an entirely new playground game.

CT CONCEPTS & PILLAR TIE-IN	TOTAL TIME REQUIRED	TECH NEEDS
Computational Thinking: • Pattern recognition • Abstraction **Pillar Alignment:** • Healthy Living	**Time required: 60 min** • Part 1: 15 min • Part 2: 20 min • Part 3: 25 min	None

PITCH YOUR PASSION

PITCH YOUR PASSION

ACTIVITY INSTRUCTIONS

Introduction

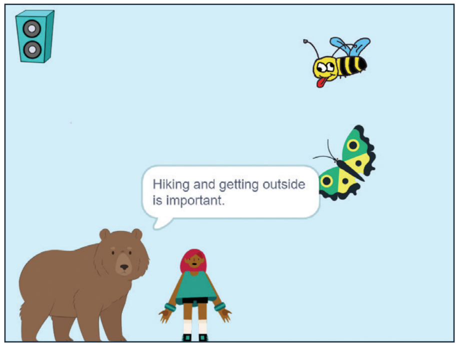
Pitch Your Passion is an introductory coding activity that teaches kids how to use code to create an animated pitch about why people should care about one of their favorite topics. Giving young people a voice to express who they are is a founding value of 4-H. In Pitch Your Passion, kids will be empowered to make decisions and gain confidence in relaying their message to others. These critical civic engagement skills help young people learn to advocate for things they care about and grow into true leaders as they build a sense of compassion, confidence and pride.

For younger kids and teens with limited coding experience (or in scenarios where you have limited time to facilitate), the first activity encourages youth to pick any topic they care about — whether it's why they think dogs should be allowed to run for president, or why they think people should care about limiting trash in the ocean — and teaches them how to create an animated pitch using code.

An optional second activity allows teens and kids with more coding experience to conduct research on a topic of interest and use code to make a compelling case for why people should care through the creation of a Public Service Announcement (PSA). The activity is recommended for kids in 6th grade and older, or for kids with more coding experience, and can easily be integrated into classroom lessons and other existing curricula. Kids can complete the activity alone or in small groups. You can choose to do either activity or both, depending on your kids' interest, age and time allotted.

These activities can be completed online or offline, but both options require access to a computer. **Follow the steps in this Online Guide to complete the activity with an internet connection. The Offline Guide is located on the included USB drive.**

Pitch Your Passion: At-A-Glance



In this activity, kids use code to animate a passion, issue or cause they care about. By animating interactions between characters, changing their environment and adding sounds, supporters, movement and more, their project will help persuade others to care about their passion as well.

This activity introduces kids to computer science and the programming language, Scratch. Kids will use different Scratch blocks to create their own unique programs.

Pitch Your Passion, Open Project Details: g.co/csfirst/NYSDkit
Pitch Your Passion Example project: bit.ly/NYSDExample1
Pitch Your Passion Example project 2 (PSA): bit.ly/NYSDExamplePSA

Goals, Objectives and Outcomes

Pitch Your Passion will help kids get comfortable with coding. Scratch is an introductory coding language designed to get kids creating, having fun and feeling confident about coding skills quickly. With just a few blocks and clicks, they can make a "sprite" (character) dance, talk or come to life in endless ways. Additionally, the computer science concepts used in Scratch can be applied to other advanced programming languages, like Python or Java.

By the end of this activity, kids will:

- be familiar with the Scratch block-based programming language;
- have learned important computer science concepts, like events, sequencing, conditionals and loops; and
- have created an animation project in Scratch.

Full Activity Time

Pitch Your Passion Activity 1: 30-60 minutes

Pitch Your Passion Optional Activity 2: 60-75 minutes

Materials to get started

- Completion Certificates: 1 certificate per student (optional, found in the Youth Workbook and on the USB drive).
- Headphones (recommended but not required).
- Pitch Your Passion: Scratch Public Service Announcement worksheets (needed for the Optional Follow-Up Activity, found in the Youth Workbook and on the USB drive).

ONLINE VERSION MATERIALS	OFFLINE VERSION MATERIALS
• Computer with internet access (recommend 1 per student or group of 2-3) • Online version of the facilitator guide (this document) • CS First website: g.co/csfirst/NYSDpitch	• Offline Activity Guide for Pitch Your Passion (located on USB drive) • Offline version of Scratch (located on USB drive) • Computer(s) with Scratch desktop installed, 1 per student or group of 2-3 • Scratch Activity Cards

IMPORTANT VOCABULARY

Code: The instructions in a computer program that computer scientists use to tell a computer what to do.

Conditional: A type of statement that tells you what to do based on the answer to a question, usually shown in programming languages with words like "if," "then" and "else." For example, conditionals could be used to specify different actions within a game: If tagged, then you are "it."

Event: Something that causes an action and can be triggered by key presses or messages sent from one part of a computer to another. For example, kids may change the color of a letter or change the size of a letter by using an event to initiate the action.

Sequencing: Putting things in order. When writing code, it's important to carefully decide the order in which the code will run. For example, kids may create a conversation between two letters by specifying which letter speaks first and which letter responds.

Loops: A way to repeat an instruction or set of instructions. For example, kids may change how long a letter spins in a circle or jumps up and down by specifying the duration of the loop.

Setup Steps for Online

1. Decide if you will be using the online or offline version for this activity and follow the corresponding setup steps.
   - If you're using the online version, proceed with the facilitator setup and instructions here.
   - If you're using the offline version, download the Scratch offline editor from **https://scratch.mit.edu/download** (or your USB drive) and use the offline lesson plan (also on your USB drive). You can also download all the activity videos in bulk from **g.co/csfirst/pitch-videos**, and/or use the custom Scratch cards.
2. Decide whether you want to create a CS First class (recommended), or get started right away without creating a class. Instructions for both options can be found below.
3. Set up your computers and workspace.
4. Read the "Activity Introduction" and "Activity Instructions" aloud and present an Example Project (**bit.ly/NYSDExample1**).

Facilitator Setup - Online Version

Go to:

g.co/csfirst/nysdkit

Review the activity and familiarize yourself with the videos and lessons.

Setup

Computers, an internet connection and optional headphones.

Students get started

Provide students with the specific URL so they can get coding.

Create

Students use logins to share and save their work.

FACILITATOR TIPS

Before the Activity:

Don't have a computer or headphones for each student? Here are some ways you can still use CS First:

- Pair or group kids. Assign one student as the "driver" who controls the computer and one as the "instructor" who describes what to do. Switch roles every five minutes.
- Whole class. Project the activity and videos on a screen where all kids can see. After watching the Introduction video, have the class suggest how you might build the project in Scratch.
- Station rotation. If you have a computer station in your classroom or club space, allow kids to rotate to the computers to complete the activity. For the rest of the kids, consider using the unplugged activities in this guide. Other resources are available on **https://www.csunplugged.org**.
- Internet connectivity issues? If you experience issues with connectivity (either unexpected or regularly), use the offline version of the activity. You can also consider downloading the Scratch Offline Editor (**https://scratch.mit.edu/download**) on each computer or pre-downloading all of the videos (available at **g.co/csfirst/pitch-videos**) so you can project them at the front of the room.

Facilitators have two options for getting started with the online version. Decide which works best for you!

CREATE A CLASS

A CS First class allows you to see the students' work, track their progress, generate Scratch usernames to save projects and more. Try to create your class before your event so you don't lose time.

Note: You need a Google account to create a class.

1. Create a class on the CS First website, g.co/csfirst/sign-in. A CS First 'class' is a club session that you're running with a given activity and group of kids. You can create a new class for every unique group of students, or use the same class code for various clubs or groups of students.
2. Visit your CS First class at g.co/csfirst/my-clubs and locate your class code for this class.
    a. This code is unique to your class. If you set up additional CS First classes, you will receive a new code for each.
    b. Students (new and returning) will need to enter this code at the beginning of class.
3. Write the following on a board or piece of paper to share — somewhere the kids can see!
    a. This URL: g.co/csfirst/go.
    b. Your unique class code.
4. Direct kids to visit g.co/csfirst/go to sign in.
5. Students will click "enter class code" and type in the unique class code.
6. When asked if they need a username and password, click "Yes." This will create one for them.
    a. For privacy reasons, CS First does not store student names on the website. We recommend you write down usernames for each student so you can keep track.
7. Hand out the Youth Workbook and encourage students to write down their username and password on the first page. They will need these to sign in to CS First and Scratch.

Note: If you forget or lose your class code, you can access it by logging on to https://csfirst.withgoogle.com/dashboard/clubs.

GO DIRECTLY TO THE ACTIVITY

This bypasses creating a class and allows kids to immediately start watching videos and creating their projects.

Kids will not be able to share or save their work unless they have an existing Scratch account, or have a valid email address that will allow them to create a new Scratch account.

1. Direct participants to visit g.co/csfirst/NYSDpitch.
2. Students will land on the activity page, watch the videos and start working on their projects.

Practice: Do an activity as a student so you have a better understanding of what your students will be creating.

READ ALOUD

PILLAR TIE-IN

Online Activity Scripts

Activity Introduction

In this activity, you will use code to create an animated story to convince people to care about a topic you're passionate about. First, you'll choose a topic. Think about something, some place or an action that you care a lot about and that you want other people to care about, too.

Here's an example of a project you can make! Visit bit.ly/NYSDExample1.

Once you've picked your topic, you'll need to decide how to tell your story in a way that will make other people want to support your cause. That's called a pitch! Learning how to pitch is an important life skill. In business, people may pitch ideas to win new business or clients, start a new project or secure funding to pursue a new idea. Outside the office, people use pitching skills to inform and persuade others to take an interest in topics they care about. You can use your pitching skills to improve your community, engage in democracy, advocate for yourself and much more.

To build your pitch you will use the programming language Scratch. When you program, or code, you provide instructions for the computer to follow. Many programmers write code in text, meaning that they type it out on the keyboard. With the Scratch language, you code using blocks that snap together like puzzle pieces.

Use only one of the following activity instructions, depending on whether you set up a class or will be going directly to the activity.

READ ALOUD

Activity Instructions (with a CS First class)

1. First, open an internet browser and go to g.co/csfirst/go to get started.
2. Click "Enter Class Code" and type in your unique class code. Click "Yes" when it asks you if you're doing Pitch Your Passion for NYSD.
3. When asked if you need a username and password, click "Yes." Then, write down the username and password the computer has generated for you, and click "Y" to confirm your CS First activity and location.
4. Now you're logged in! You will see the "Pitch Your Passion" activity page. Plug in your headphones if you have them, and watch the introductory video at the top.
5. Open the starter project linked next to the video. This will open Scratch in a new tab.
6. Then, return to CS First to select an add-on video with a new coding challenge, and follow the instructions.
7. If you want to share and save your work, remember to click "Sign in" on the Scratch website. Sign in using the same username and password. Once you have signed in, you can remix, save and share your project.

Activity Instructions (without a CS First class)

1. Open an internet browser and go to g.co/csfirst/NYSDpitch to get started.
2. You'll see the "Pitch Your Passion" activity page. Plug in your headphones if you have them, and watch the introductory video at the top.

3. Open the starter project linked next to the video. This will open Scratch in a new tab.

4. Then, return to CS First to select an add-on video with a new coding challenge, and follow the instructions.

FACILITATOR TIPS

During the Activity:

- Check in with kids as they watch the videos and begin their projects.
- At the conclusion of the intro video, kids should open Scratch in a new tab.
- If you notice a student watching add-on videos without a Scratch tab open, instruct them to open Scratch, then they can return to the add-on video.
- Leverage the expertise of your students. Instead of answering student questions directly, open it up to the class to see if others have suggestions, solutions or a workaround.
- On all CS First videos, written transcripts, closed captions and playback speed (Slow, Normal and Fast) are available options to students. These features may help kids absorb the information in a way that better aligns with their learning style.
- Use the add-on solution guide in this guide, or pass out Scratch cards, to help kids who get stuck.

Discussion Questions

- Can you show me what you've created so far?
- What blocks are you using?
- What did you learn when you watched this video?
- How would you explain the code in your project to a younger student or sibling?

Prompts to encourage sharing and troubleshooting:

- What's something else that you could potentially do to this sprite or backdrop?
- If I changed [choose a value or block] to [choose another value or block], what do you think would happen? Let's test that hypothesis. What happened?

Wrap-up

- When there are five minutes left, remind students to click the green "next" button to complete a short survey. Give them 3-4 minutes to complete the survey.
- Have kids share their projects with a neighbor or do a whole gallery walk.
- If you created a class, instruct students to share their project in Scratch. This will allow you to view their projects from your Student Stats page. To do this, instruct the kids to go to the project in Scratch and click Share in the top right corner. If the Share button is not visible, the student should click Remix, then Save, then Share.

READ ALOUD

Reflection Questions:

Let's discuss what you learned during this activity:

- What topic did you pitch in your program? What inspired you to make it?
- Have you done any research about this topic before?
- If you had more time, what would you add to your project? How would you do it?
- What was your favorite part of this activity?
- Why do you think others will be convinced to care about your project?
- What did you learn about coding?
- What was the most challenging part of this activity?

Take It Further Activity (optional): Scratch Public Service Announcement

This activity allows kids to conduct research on a topic of interest and use code to make a compelling case for why people should care. Learning how to create a persuasive argument and communicate the right message for listeners is an important part of developing a strong PSA. The basic research and persuasive messaging skills kids learn in this activity will help prepare them for the leadership roles they will take on in the future. The activity is recommended for kids 6th grade and older, or for kids with more coding experience, and can be easily integrated into classroom lessons and other existing curricula. Kids can complete the activity alone or in small groups.

Setup Steps

Use the same setup instructions as the first activity. If kids created usernames and passwords while completing activity one, they should use these to log in and save their work for activity two.

Activity Instructions

As the facilitator, you should guide kids through steps 1 and 2. Kids can use the Youth Workbook to complete steps 3-6.

READ ALOUD

In this activity, you'll create a Scratch program that's a "Public Service Announcement" (PSA). A PSA is a message that raises awareness about a topic or social issue; it motivates people to take positive action for that cause. A PSA grabs the audience's attention, sends a simple and clear message, and encourages people to consider changing a behavior or their attitude towards an issue.

As a group, we'll watch a few example PSAs and discuss their styles and impact. Then you'll use the steps outlined in the Youth Workbook to help you develop your PSA.

Step 1: Watch a PSA

Show a few of the following Scratch or YouTube examples to the group:

Scratch Projects:

- Adopt don't shop by jeilanii (**bit.ly/NYSDPSA1)**
- Halloween Culture ≠ Costume by amee- (**bit.ly/NYSDPSA2**)
- Clean water by Echostrike (**bit.ly/NYSDPSA3**)
- Google is not a source by DaMan56100 (**bit.ly/NYSDPSA4**)

Videos:

- Award Winning Anti-Smoking Commercial (**bit.ly/NYSDPSA5**)
- Hearing Loss PSA Challenge Videos (**bit.ly/NYSDPSA6**)

READ ALOUD

Step 2: Discuss

Now that you've seen some PSAs, let's discuss the following questions:

- What social issue or message was this PSA trying to address?
- What should the viewer do, the "call to action," after they've heard the message?
- Who was the intended audience of each PSA?
- What was the tone and "persuasive tool" used to communicate the message? (Comedy? Statistics? Appeal to emotions?)
- What did the PSAs have in common?
- What did the creators use to communicate their message?

- Were you persuaded? Why or why not?
- Make a list: If you could change anything about the PSA, what would you do to make it better?

Use the Youth Workbook to complete Steps 3-6 as you research, plan and code your Scratch Public Service Announcement.

Step 3: Choose your own topic

Kids select a topic or issue that is important to them.

1. Create a list of possible topics related to the issue chosen.

   Examples:

   - Is the focus on supporting an organization? List some actions that the organization does or goals they try to achieve.
   - Is the focus on preserving and protecting the environment? Sub-topics could include protecting wildlife, reducing litter or using solar power.
   - Is the focus on the body and being healthy? Think about issues that speak to the general idea, like healthy foods or hearing loss.
2. Encourage students to share ideas in a small group and discuss why it's important.
3. They should write down other opinions and other group members' perspectives related to the topic chosen.

   TIP: Sometimes others' contrary opinions can be helpful starting points for research to persuade them.
4. Choose a specific topic for the passion project. Encourage specificity.

Step 4: Research your topic

Using the internet and/or other resources available (e.g. newspapers, books, other people), kids research the topic that they've chosen and find data or statistics that will help convince their audience why their topic is important.

EXAMPLE:

A PSA topic to encourage people to go outdoors and hike might use the following data/information: "On average, children aged 10 to 16 now spend only 12.6 minutes a day on outdoor activity compared to 10.4 hours being motionless." Source: UK Study sponsored by National Trust.

Step 5: Plan

Students select a topic or issue. It could be about a place they want to help and protect, a group or organization, or other issues like online safety or being healthy.

Kids plan a national (or global) campaign with one or more different solutions to solve their problem or advocate for their issue. Think about how you would create a PSA on your topic.

- What words could you use to convey your message or goal?
- Is there a catchy phrase or slogan you can use?
- What would you say to share out the importance of your message?
- How do you convince or persuade people?
- How will you share your facts? Will it be funny, or dramatic? Show positive outcomes or negative ones?

Step 6: Create your own PSA Scratch Project

Kids use Scratch to create a PSA and share it with the world. The suggested PSA structure includes an overview of the topic, incorporating facts into the body of the project, adding a personal story or narrative, closing the project with a persuasive statement for the audience to take action, and including credits that list the sources and references where they found their information.

FACILITATOR TIPS

Youth can use the Scratch coding challenges from the first Pitch Your Passion activity to create their PSA. In particular, they may find the following add-ons to be helpful:

- Speak out.
- Change scene.
- Add background music.
- Include supporters.
- Make some noise, use your voice.

New add-on challenges that can bring their PSA project to life:

- Broadcast.
- Weather/confetti: Program snow, rain, confetti or cheese-puffs to fall from the sky.
- Supportive phrases (using “data”): Program a sprite to say a supportive phrase from a list.
- Make an entrance: Program your main character to spin onto the stage.
- No-code add-on: Add instructions, links and tips on the Project Page of the program; have students exchange, collaborate and/or add comments.

Credits for this activity:
Scholastic: **bit.ly/NYSDcredit1**
Read Write Think (RWT): **bit.ly/NYSDcredit2**
CTE Online: **bit.ly/NYSDcredit3**

Kids are encouraged use the following structure to map out their PSA.

1	**Overview.** The first part of your project should have an opening message that grabs the audience's attention. Use leading questions, like… "Did you know…", "Would you believe that…" or a statistic.
2	**Facts.** The middle part of your project should give the audience facts about why your topic is important. Use the research you collected about your topic. Include facts and their sources.
3	**Personalization.** You can also weave in a personal story or narrative about why you (or the character/narrator that you've chosen) cares about the topic.
4	**Closing.** Close your project by persuading the audience to take action.
5	**Credits.** List credits to your project, using sources and references for where you found your information. If you'd prefer not to use a scene in your Scratch project on credits, make the credits a part of your project description when you save and share your work.

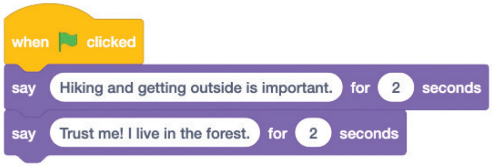
Add-on Solution Guide

Use this guide as a reference during the activity to see what the code might look like for each add-on coding challenge. The description tells you the outcome of each add-on video. The images are example code blocks that students can add in Scratch. **Note:** The Add-on Solution Guide contains example code for both activity add-ons: Activity 1 - Pitch your Passion and Activity 2 - Scratch Public Service Announcement.

Activity 1 — Add-on Code: Pitch your passion

Speak out

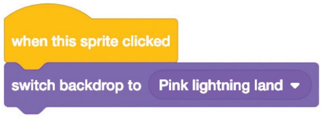
Make a character talk about the issue chosen.



```
when ⚑ clicked
say (Hiking and getting outside is important.) for (2) seconds
say (Trust me! I live in the forest.) for (2) seconds
```

Change scene

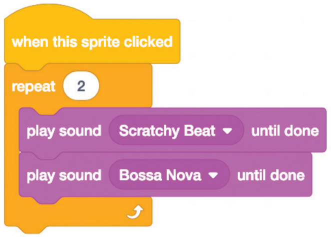
Change the animation scenery by adding more backdrops.



```
when this sprite clicked
switch backdrop to (Pink lightning land ▾)
```

Add sound effects

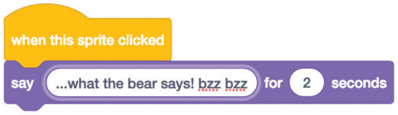
Add or record sound to accompany the project and excite the audience.



```
when this sprite clicked
repeat (2)
    play sound (Scratchy Beat ▾) until done
    play sound (Bossa Nova ▾) until done
↺
```

Include supporters

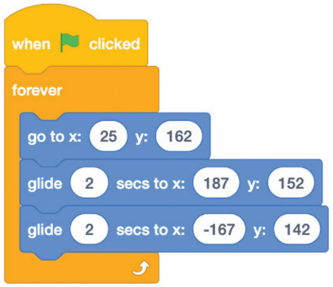
Add or draw more characters who support your cause.



```
when this sprite clicked
say (...what the bear says! bzz bzz) for (2) seconds
```

Move around

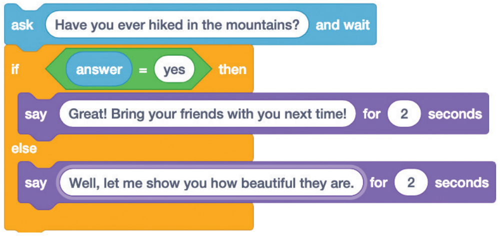
Make your characters move to animate parts of your program.



```
when ⚑ clicked
forever
    go to x: (25) y: (162)
    glide (2) secs to x: (187) y: (152)
    glide (2) secs to x: (-167) y: (142)
↺
```

Question and answer

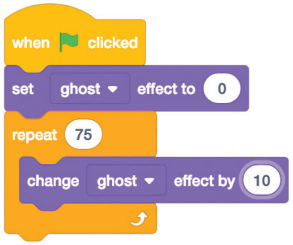
Ask the audience a question and make something happen based on the response.



```
ask (Have you ever hiked in the mountains?) and wait
if <(answer) = (yes)> then
    say (Great! Bring your friends with you next time!) for (2) seconds
else
    say (Well, let me show you how beautiful they are.) for (2) seconds
```

Add a title

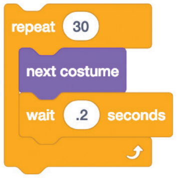
Create and design a title slide for your passion project so the audience knows what the project is about.



```
when ⚑ clicked
set ghost ▾ effect to 0
repeat 75
    change ghost ▾ effect by 10
```

Change costumes

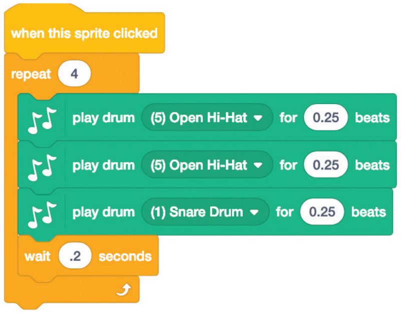
Animate your characters by making them change costumes.



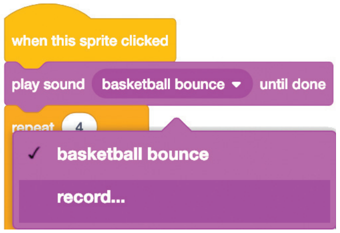
```
repeat 30
    next costume
    wait .2 seconds
```

Make some noise, use your voice

Make a unique beat, record sounds and use your own voice in the project.



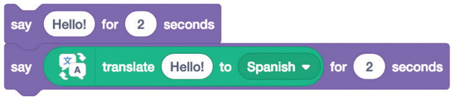
```
when this sprite clicked
repeat 4
    play drum (5) Open Hi-Hat ▾ for 0.25 beats
    play drum (5) Open Hi-Hat ▾ for 0.25 beats
    play drum (1) Snare Drum ▾ for 0.25 beats
    wait .2 seconds
```



```
when this sprite clicked
play sound basketball bounce ▾ until done
repeat 4
```

Translate message

Share your passion project with people who speak other languages.



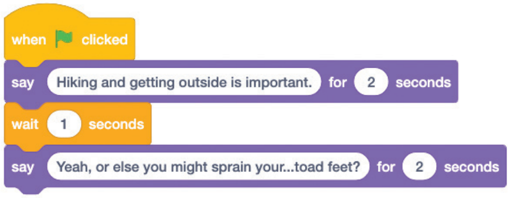
```
say Hello! for 2 seconds
say translate Hello! to Spanish ▾ for 2 seconds
```

Activity 2 — Add-on Code: Public Service Announcement (PSA)

Engage supporters

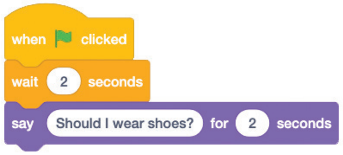
Make two characters talk to one another about the issue you've chosen.

Sprite 1



```
when ⚑ clicked
say Hiking and getting outside is important. for 2 seconds
wait 1 seconds
say Yeah, or else you might sprain your...toad feet? for 2 seconds
```

Sprite 2

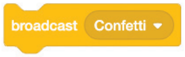


```
when ⚑ clicked
wait 2 seconds
say Should I wear shoes? for 2 seconds
```

Weather or confetti

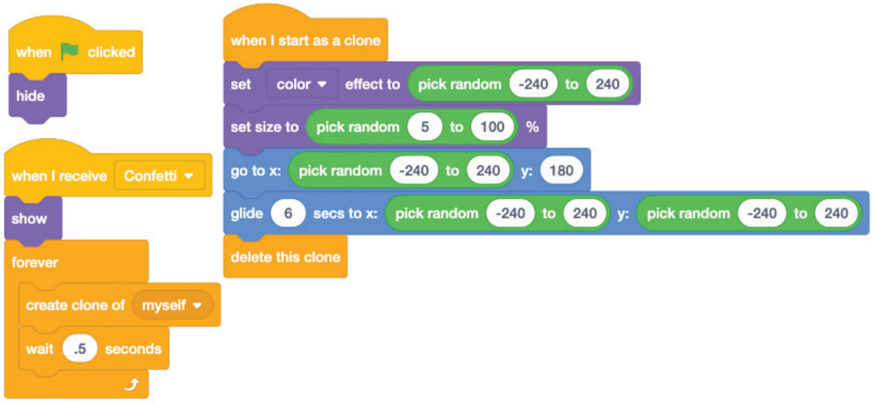
Program confetti, snow, hearts or anything you like to fall from the sky.

Sprite 1



```
broadcast Confetti ▾
```

Sprite 2

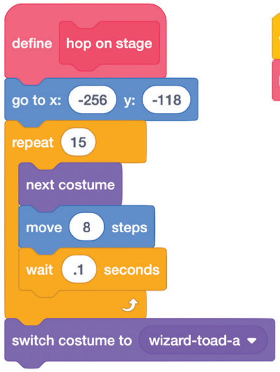


```
when ⚑ clicked
hide

when I receive Confetti ▾
show
forever
    create clone of myself ▾
    wait .5 seconds
↻

when I start as a clone
set color ▾ effect to pick random -240 to 240
set size to pick random 5 to 100 %
go to x: pick random -240 to 240 y: 180
glide 6 secs to x: pick random -240 to 240 y: pick random -240 to 240
delete this clone
```

Come on stage

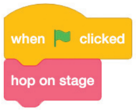
Program the main character to come on stage with an animated motion effect.

Define the motion



```
define hop on stage
go to x: -256 y: -118
repeat 15
    next costume
    move 8 steps
    wait .1 seconds
switch costume to wizard-toad-a ▾
```

Add to sprite

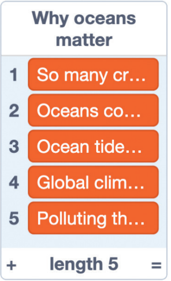


```
when ⚑ clicked
hop on stage
```

Supportive phrases

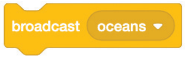
Program the supporters (other character sprites) to say supportive phrases from a list of options about the issue or passion.

Create variable list



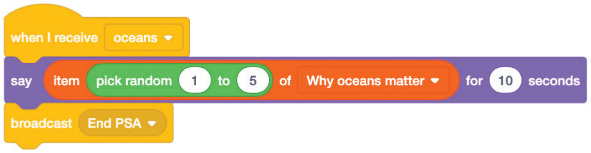
Why oceans matter	
1	So many cr…
2	Oceans co…
3	Ocean tide…
4	Global clim…
5	Polluting th…
+	length 5 =

Define broadcast



```
broadcast oceans ▾
```

Activate supporter

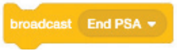


```
when I receive oceans ▾
say item pick random 1 to 5 of Why oceans matter ▾ for 10 seconds
broadcast End PSA ▾
```

Broadcast finale

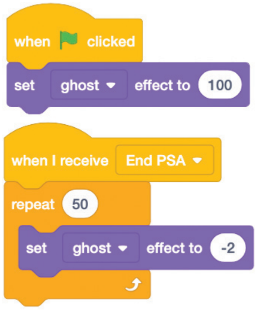
Broadcast a message to show during the end/credits of the project.

Define event to broadcast



```
broadcast End PSA ▾
```

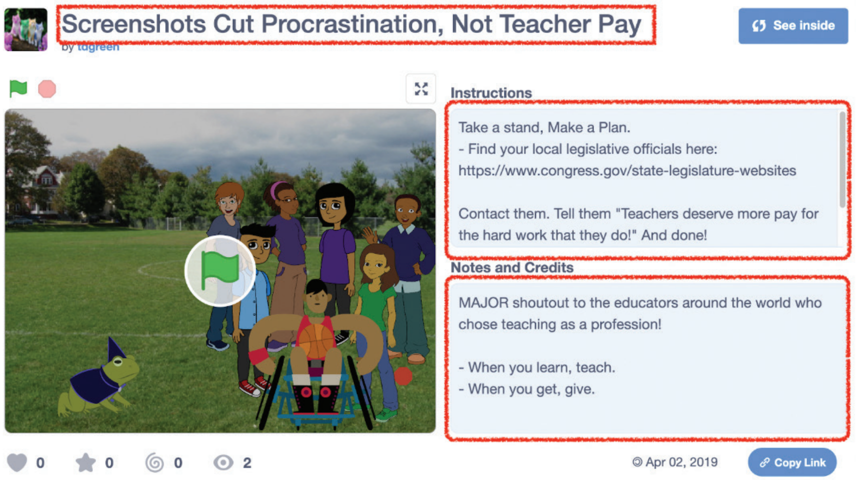
Program your final slide



```
when ⚑ clicked
set ghost ▾ effect to 100

when I receive End PSA ▾
repeat 50
    set ghost ▾ effect to -2
```

Take action

Share project and add call-to-action details and info. Include prompts and credits.



Screenshots Cut Procrastination, Not Teacher Pay

by tdgreen

Instructions

Take a stand, Make a Plan.
- Find your local legislative officials here:
https://www.congress.gov/state-legislature-websites

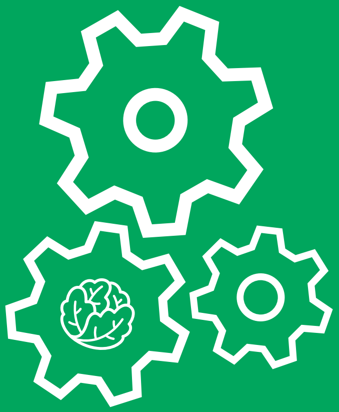
Contact them. Tell them "Teachers deserve more pay for the hard work that they do!" And done!

Notes and Credits

MAJOR shoutout to the educators around the world who chose teaching as a profession!

- When you learn, teach.
- When you get, give.

C#
100110



HACK YOUR HARVEST

HACK YOUR HARVEST (UNPLUGGED)

ACTIVITY INSTRUCTIONS

Introduction

In this activity, you'll introduce kids to the concepts of automation and efficiency, and teach the basic principles of writing instructions for computers to follow. Part 1 challenges kids to find efficient solutions to puzzles that simulate programming an automated tractor. In Part 2, kids are introduced to a popular computer science problem, the traveling salesman problem, through a similar puzzle format. In Part 3, kids design their own puzzles and have fellow classmates try to solve them.

Goals, Objectives and Outcomes

By the end of this activity, kids will:

- understand the importance of optimal efficiency as it applies to writing computer code;
- understand the terms 'automation', 'program', 'programming' and 'programming language' as they apply to computer science;
- have designed programs for a board game that simulate an automated tractor; and
- learned about the traveling salesman problem, a mathematical problem in which one tries to find the shortest route that passes through each city on a map.

Full Activity Time: 45 minutes

Part 1: Get to the Barn (15 minutes)

Part 2: The Traveling Tractor (15 minutes)

Part 3: Create Your Own! (15 minutes)

Materials

- 5 Hack Your Harvest game boards, one per group.
- 5 dry erase markers with eraser caps.
- 5 sets of game pieces (5 boulders, 5 brambles, 5 heads of lettuce and 1 tractor per group of two).
- Hack Your Harvest Rules Guide (found in the Youth Workbook).

Not included in the kit:

- 5 pencils; one per group.

IMPORTANT VOCABULARY

Automation: Having computers or machines do repetitive tasks.

Optimal Efficiency: Achieving maximum productivity with minimum waste.

Program: A set of instructions that tells a computer how to perform a specific task.

Programming: The action or process of writing computer programs.

Programming language: A set of specific instructions used by computer scientists to tell computers what to do.

Traveling Salesman Problem: A mathematical problem in which one tries to find the shortest route that passes through each city on a map.

Steps

1. Split kids into groups or pairs (depending on numbers). Kit materials are provided for five groups of two.
2. Read the Activity Introduction and Youth Instructions sections on the next page out loud to the class.
3. Pass out the materials.
4. Show or point out the Hack Your Harvest puzzle setups, code and legend to the class (they also appear in the Youth Workbook). Check for understanding of each.
5. Pause after each puzzle to give groups time to complete it. Instruct groups not to move on to the next puzzle unless instructed to do so. Most puzzles are followed by a Facilitator Read Aloud section that allows for discussion and reflection.
6. Provide time at the end for kids to create their own puzzles and have them tested by their peers.
7. Facilitate the Reflection Questions section at the end of the activity.

Activity Introduction and Youth Instructions

READ ALOUD

Prompting Question

What comes to mind when you hear the term "efficient" or "efficiency"?

Optimal efficiency means to achieve maximum productivity with minimum waste. There are different kinds of efficiency depending on what kind of "waste" you are concerned about. For example, building a car efficiently might mean conserving materials to use the smallest amount of metal and plastic, or it could mean utilizing an assembly line to reduce the total time it takes to build a large number of cars. A given task can be efficient in one way but inefficient in another.

Can you think of an example where this might be the case?

Possible Examples:

- Hiring more workers might result in a project being completed in less time, but costs more money.
- Walking or riding a bike to school is healthier and better for the environment than driving, but can take longer.

Efficiency is important in CS because computers have limited resources, including time and memory. To ensure they do not overload a computer and that they get results back in a reasonable amount of time, computer scientists must write efficient programs.

READ ALOUD | PILLAR TIE-IN

Efficiency is also important in agriculture, one of the 4-H pillars. As the population of Earth increases, farmers need to harvest more crops and raise more livestock to feed people. The Earth has a limited amount of space on which to farm, however, and farmers need to keep the cost of food production low. This means they need to be as efficient as possible, both with the space they are using and the amount of money it takes to produce food. What a tricky problem!

One way farmers keep production costs low is by using automation. ***Automation*** means having computers or machines do repetitive tasks. Many farmers are now using automated harvesting equipment.

(*DO NOT READ THIS 'OPTIONAL' SECTION ALOUD)

Optional: Play the following four minute video which shows how computer science is revolutionizing agriculture: bit.ly/NYSDagriculture

READ ALOUD

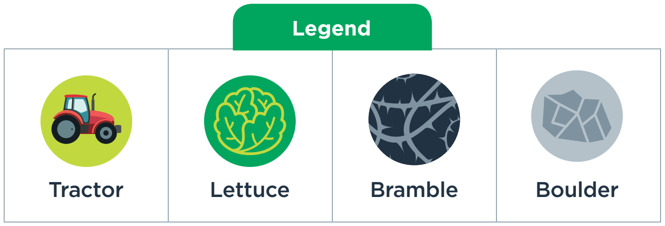
The action or process of writing computer programs is called ***programming***. In the following activities, we're going to simulate farmers programming an automated tractor. We'll write efficient programs that make the tractor complete the task using the fewest possible instructions.

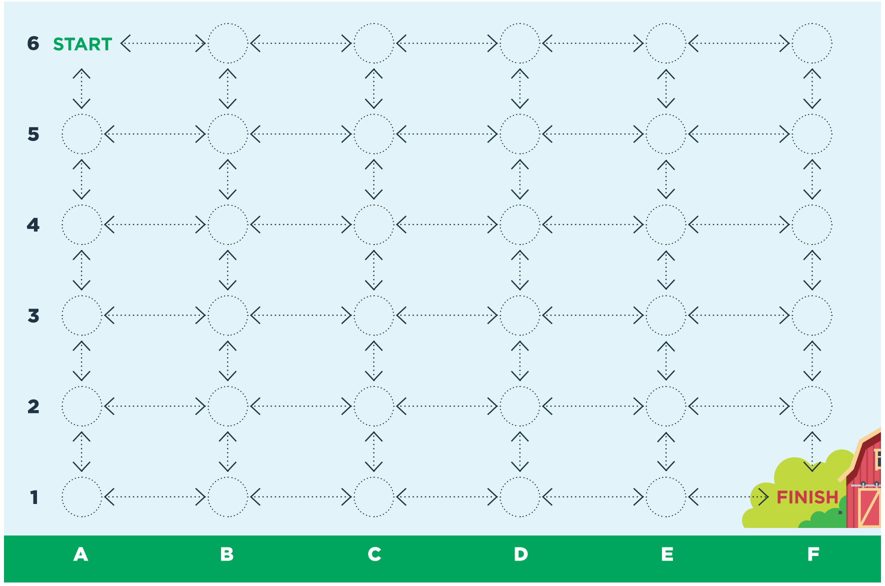
READ
ALOUD

Part 1: Get to the Barn

To make sure computers work as expected, computer scientists use a **programming language** to tell computers exactly what to do. In the following puzzles, you'll need to program the tractor to get to the barn. You'll be using a special set of instructions, shown in your Youth Workbook, to move the tractor up, down, left or right (not diagonal) one spot per instruction. Each arrow instructs the tractor to move one spot. You want to write a program that uses the fewest possible instructions. Go ahead and set up your board according to the Puzzle 1 setup and then solve it.

Puzzle 1 Setup





Puzzle 1 Solution

1	2	3	4	5	6	7	8	9	10
→	→	→	→	→	↓	↓	↓	↓	↓

FACILITATOR TIPS

There are multiple correct solutions to all of the puzzles. The solutions in this guide are just examples of correct ones.

READ ALOUD

Reflection Questions:

You just wrote a ***program*** because you used an agreed upon set of instructions to tell a device how to do a task. How many of you solved the puzzle using 15 instructions or fewer?

(Allow kids to raise their hands.)

Great job! Can anyone tell me what the minimum number of instructions needed to complete this puzzle is?
10 is the best possible (see the solution).

FACILITATOR TIPS

If anyone responds that they got a value lower than 10, there is an error. Have them compare with a neighbor to check each other's logic.

READ ALOUD

Was there more than one solution to this puzzle that used the minimum number of instructions?

(Allow kids to raise their hands).

Yes (5 right, 5 down; 5 down, 5 right; many kinds of zig-zags)

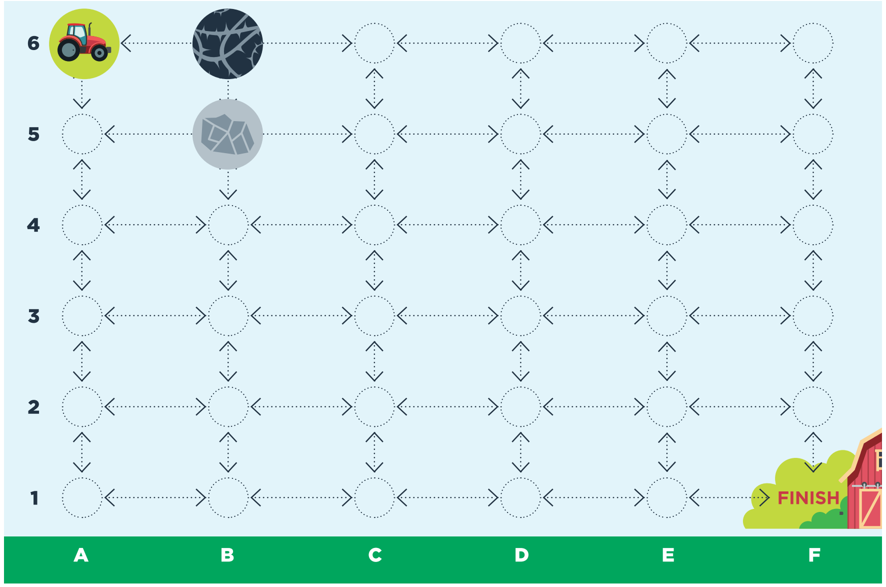
It is often the case that there is more than one efficient solution to a problem, just like there can be more than one right way to get the solution to a math problem. The solution you choose can depend on the context. We'll see an example of this in the next puzzle.

READ ALOUD

In Puzzles 2 and 3, we're going to add new components to our game board. This time, there will be two kinds of obstacles in the tractor's way: bramble and boulders.

Your tractor is never allowed to enter a spot with a boulder on it, however it is equipped with a dozer blade that will allow you to clear the brambles. We're going to add a new element to our programming language to deploy our tractor's dozer blade. Before you're allowed to move onto a spot with a bramble, you must deploy the dozer blade by using the * instruction. For example, in Puzzle 2 in your workbook, to move right one spot you would need a * in instruction 1 and a "➔" in instruction 2. As you can see, moving onto a spot with a bramble now requires two instructions instead of just one. You'll now have to determine whether it is more efficient to bulldoze the bramble or go around it. Go ahead and solve Puzzle 2.

Puzzle 2 Setup



Puzzle 2 Solution

1	2	3	4	5	6	7	8	9	10
↓	↓	↓	↓	↓	→	→	→	→	→

READ ALOUD

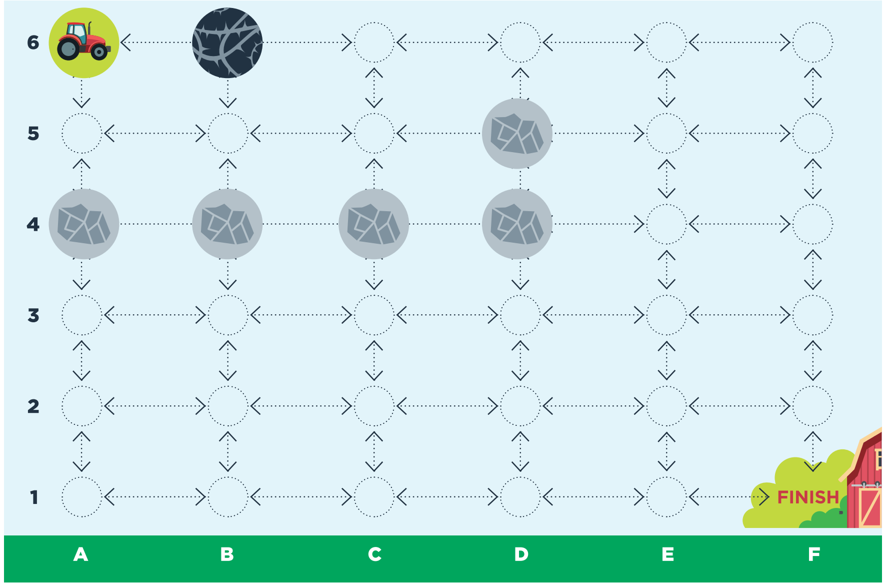
Reflection Questions:

What was the minimum number of instructions needed to solve this puzzle?
10 (see the solution above).

Was it more efficient to bulldoze the bramble or go around it?
Go around it.

Notice that the efficient solution for this puzzle can also be used as an efficient solution for Puzzle 1. If we were trying to find one solution that worked for both we might use this one. Now go ahead and solve Puzzle 3.

Puzzle 3 Setup



Puzzle 3 Solution

1	2	3	4	5	6	7	8	9	10	11
*	→	→	→	→	→	↓	↓	↓	↓	↓

READ ALOUD

Reflection Questions:

What was the minimum number of instructions needed to solve this puzzle?

11

Was it more efficient to bulldoze the bramble or go around it?

Bulldoze

Can you think of an instance in the real world where you might choose to go around an obstacle instead of bulldozing it, even though it's more efficient to remove it?

If excavating it is harmful to the environment or if there is an obstacle like a tree that you'd like to keep.

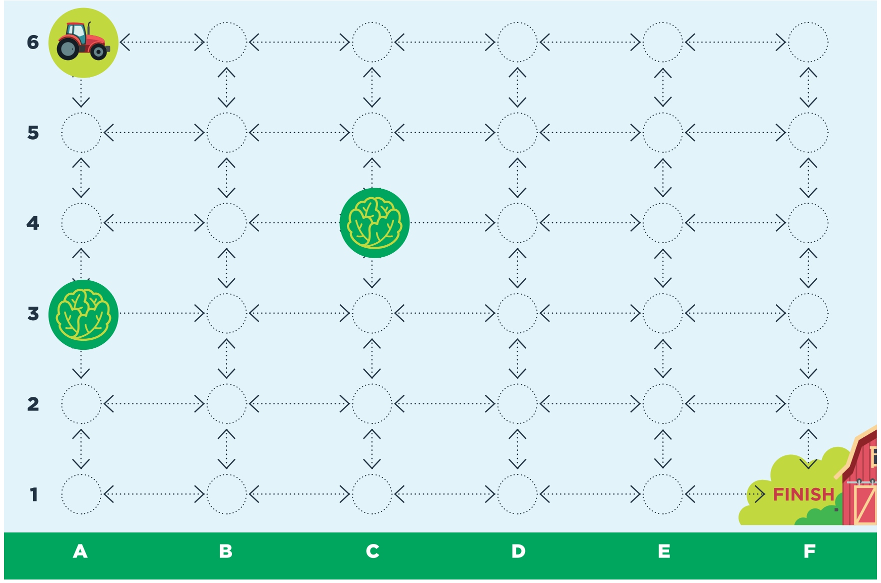
READ ALOUD

Part 2: The Traveling Tractor

In Puzzles 4 and 5, lettuce is added to the board. To collect a head of lettuce, we're going to add a new element to our programming language. Use the $ instruction *after* you've moved your tractor to the spot where the lettuce is to collect it. You must collect all of the heads of lettuce before you go to the barn (Finish). You can collect the lettuce in any order you want, but the goal is to be optimally efficient.

Go ahead and solve Puzzles 4 and 5.

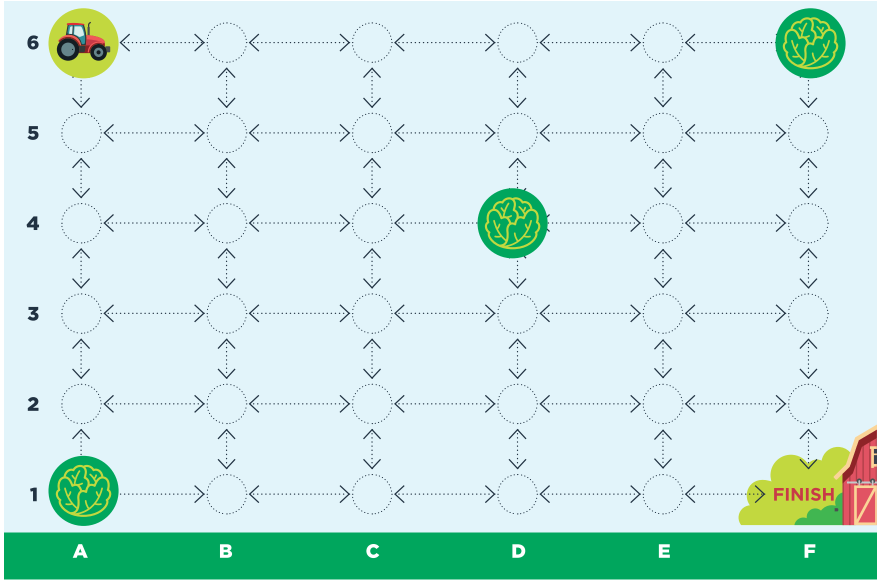
Puzzle 4 Setup



Puzzle 4 Solution

1	2	3	4	5	6	7	8	9	10	11	12	13	14
↓	↓	↓	$	→	→	↑	$	→	→	→	↓	↓	↓

Puzzle 5 Setup



Puzzle 5 Solution

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
↓	↓	↓	↓	↓	$	↑	→	↑	→	↑	→	$	↑	→	↑	→	$	↓	↓	↓	↓	↓

READ ALOUD

Reflection:

You may have noticed that this problem gets much more difficult to solve when more heads of lettuce are added to the board. In Puzzle 4 there were only 2 routes you had to check, but in Puzzle 5 there were 6 possible routes. If there were 4 heads of lettuce on the board, there would be 24 possible routes to check! With 10, there would be 3.6 million possible routes!

The general statement of this problem is called **"the traveling salesman problem."** Computer scientists are interested in problems like this one because it can take a computer an extremely long time to find the optimal solution when there are a lot of possibilities to check. This problem can be restated to apply to many real-world scenarios, such as mail delivery and airline flight planning, so it is one that computer scientists would like to be able to solve efficiently. Using some clever tricks, mathematicians and computer scientists have found ways to not have to check every possible path. Even using these tricks, however, the problem can sometimes still take a long time for a computer to solve, and computer scientists often have to settle for a "good enough" solution.

FACILITATOR TIPS

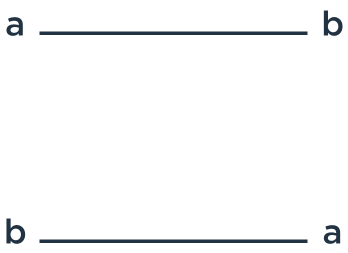
Use the optional Take It Further Discussion if youth want to know more about this problem and why adding more heads of lettuce rapidly increases the number of paths. It is recommended that you use a blackboard or piece of paper for this discussion to draw out the diagrams.

READ ALOUD

Take It Further Discussion (optional)

Let's investigate why adding new heads of lettuce creates so many additional routes. By route, we mean the order in which you pick up all the lettuce, not the path your take from lettuce to lettuce. First, we'll break down a few small cases and then we'll see if we can find the pattern. Suppose we have just one lettuce. Let's name this lettuce "a." How many different orders do we have to check? Just one: we begin at the start, get lettuce "a," and then end at the barn. Notice that we always begin at the start and end at the barn; since these are always the same we'll ignore these stops as we continue our investigation.

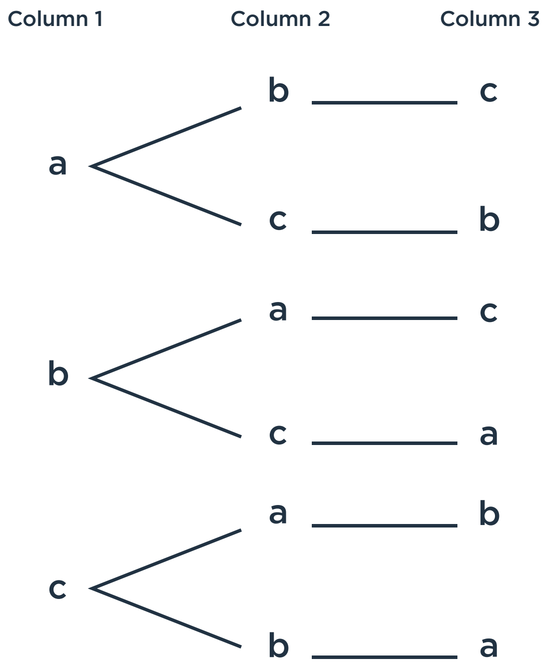
Consider the case when we have two heads of lettuce, let's name them "a" and "b." There are now two orders we have to check. Let's draw this out:



READ ALOUD

We have two choices for the first lettuce we pick up: we can either get lettuce “a” first or get lettuce “b” first. Once we have decided which lettuce to pick up first, next we then have to decide which lettuce to pick up second. If we picked up lettuce “a” first, there is only one possibility – we must now pick up lettuce “b” because it is the only one remaining. If we picked up lettuce “b” first, we must pick up lettuce “a” second. This gives us 2×1=2 total paths (that's two choices for the first lettuce we picked up multiplied by one remaining choice for the second lettuce).

Finally, let's consider the case when we have three heads of lettuce named “a,” “b” and “c.”



We have three choices for the first lettuce to pick up. Once we have made this decision, there are two remaining choices for which lettuce we pick up second. For example, if we picked up lettuce “a” first, we can either pick up lettuce “b” second or we can pick up lettuce “c” second. Once we have decided which lettuce we want to pick up first and second, there is only one remaining choice for which lettuce we want to pick up third. For example, if we pick up lettuce “c” first and lettuce “a” second, we must pick up lettuce “b” third. This gives us 3×2×1=6 total paths. Looking at the diagram, you can see three choices for the first lettuce to be picked up in column one. For each of these possible choices, there are two different possibilities for which lettuce to pick up second. This gives us a total of 3×2=6 possible orders in (column two). Finally, we must choose which lettuce we want to pick up third. For each possible way to pick the second lettuce, there is only one way to choose the third lettuce. This gives us 3×2×1 total options in the last column.

Do you see the pattern? For four heads of lettuce, we would have 4×3×2×1=24 total paths. For ten heads of lettuce, we would have 10×9×8×7×6×5×4×3×2×1=about 3.6 million possible paths! This pattern of multiplying together all of the counting numbers in a given problem is common in mathematics, statistics and computer science. The result is called a factorial; instead of writing out all of the numbers and the multiplication signs, we just write the largest number with an exclamation point afterward. For example 10!=10×9×8×7×6×5×4×3×2×1.

(*DO NOT READ THIS 'OPTIONAL' SECTION ALOUD)

If kids want to learn more about the traveling salesman problem, the following websites provide more information:

- Cool animations that show different algorithms for some small sample sizes: bit.ly/NYSDsalesman1.
- Traveling salesman problem visualization video: bit.ly/NYSDsalesman2.
- Simulation for traveling to every state capital in the contiguous US: bit.ly/NYSDsalesman3.

READ ALOUD

Part 3: Create Your Own!

Now you're going to create your own puzzles. Be sure to use no more than 5 boulders, 5 brambles and 5 heads of lettuce. Swap puzzles with someone else in your group and try to find the most efficient solutions to each other's puzzles. You should know the solution to your own puzzle before you have your partner try to solve it.

Reflection Questions:

Reflecting on the activities we completed today: what did you learn, why is it important and how might you use this information next?

FACILITATOR TIPS

For easy storage and reuse, have kids stick the decals directly onto their game boards at the end of the activity.

PROGRAM YOUR PLAYGROUND

PROGRAM YOUR PLAYGROUND (UNPLUGGED)

ACTIVITY INSTRUCTIONS

Introduction

In this activity, you'll introduce kids to the concepts of pattern recognition and abstraction. In Part 1, kids will create their own modified version of 'tag' as they learn the concept of conditionals, before taking turns playing their new games. In Part 2, kids use pattern recognition to find the similarities in various versions of tag and work in groups to create an entirely new kind of tag. Finally, in Part 3, kids abstract concepts from several examples, then use what they learned in the previous activities to create an entirely new playground game.

Goals, Objectives and Outcomes

By the end of this activity, kids will:

- use elements of the computer software design process to create and test their own games;
- understand the terms 'conditionals' and 'pattern recognition' as they apply to computer science; and
- use abstraction to identify and group different types of playground games.

Full Activity Time: 60 minutes
Part 1: Tag, You're It (15 minutes)
Part 2: Finding the Pattern (20 minutes)
Part 3: Let's Play! (25 minutes)

Materials

- 2 inflatable beach balls (one green, one yellow).
- Program Your Playground worksheets (found in the Youth Workbook).

Not included in the kit:

- Pencils, one per group.
- Optional - a variety of common playground equipment (hula hoops, balls, buckets, bats, etc.).

IMPORTANT VOCABULARY

Abstraction: Identifying and extracting relevant information to define the main ideas.

Computer Software: A collection of instructions that tell a computer how to perform a group of tasks.

Conditional: A type of statement that tells you what to do based on the answer to a question, usually shown in programming languages with words like "if," "then" and "else." For example, conditionals could be used to specify different actions within a game: If tagged, then you are "it."

Pattern Recognition: Observing patterns, trends and regularities in data.

Software Design Process: The process by which computer scientists create a new piece of software. Some parts of this process include pattern recognition, abstraction and user testing.

User Testing: The process of evaluating a product or prototype with real users.

Steps

1. Blow up the beach balls prior to the activity.
2. Split kids into two groups or more (depending on numbers). Kit materials are provided for two groups of five.
3. Read the Activity Introduction and Youth Instructions sections on the next page out loud to the class.
4. Stop between each part to give groups time to create their games.

   Walk around to check for understanding or to provide encouragement and ideas if groups are stuck.
5. Each part is followed up by a Facilitator Read Aloud section that allows for discussion and reflection.
6. Provide time at the end for groups to create and test their playground games.
7. Facilitate the Reflection Questions section at the end of the activity.

FACILITATOR TIPS

- This activity is best done in an area with plenty of space such as a playground, field or gymnasium.
- If you are confined to a smaller indoor space like a classroom, try to make as much room as you can by pushing furniture out of the way, and instead of running, have kids crawl, hop, skip or gallop. If the furniture cannot be moved, have the youth remain seated or stand in place and pass the beach ball around to represent being tagged. This option also works well when there are participants with limited mobility.
- The instructions in this activity are for a group of 10-12 kids who are divided into two groups of 5-6. If you have a larger group, split the class into four groups of 4-6 and have two groups partner to play their versions of each game rather than combining the entire class to play all four versions.
- In Part 3, kids will be designing their own playground games using the materials at hand and their own creativity. While physical materials aren't required, providing a variety of common playground equipment or toys (hula hoops, balls, buckets, bats, etc.) can add to the fun and innovation.

Activity Introduction and Youth Instructions

READ ALOUD | **PILLAR TIE-IN**

Prompting Question

Can you name some examples of how computer science is helping us become more active and healthy?

From wearable technology that measures your physical activity, stress levels and amount of sleep, to smartphone apps that remind you to drink more water or make better food choices, computer science is making it easier to maintain a healthy lifestyle. Computer scientists and biomedical engineers are also developing technologies to treat chronic conditions like diabetes and asthma.

(*DO NOT READ THIS 'OPTIONAL' SECTION ALOUD)

Optional: Show youth the following infographic which demonstrates how computer science is revolutionizing healthcare through wearable technology: bit.ly/NYSDweartech.

READ ALOUD

Each of these examples requires computer software to operate. **Computer software** is a collection of instructions that tells a computer how to perform a group of tasks. A video game is a special type of computer software. Computer scientists have even developed exercise video games that encourage users to do healthy levels of activity by making exercise fun.

Can you name some examples?

Possible Answers: Wii Fit, Dance Dance Revolution, Pokémon Go, etc.

To design these games, computer scientists first look at various exercises, sports and games that people like to play. They identify the elements that make games fun and add exercises and movements that help you get in shape. Using all of this information, software developers then create a new exercise video game. All of these steps are part of the **software design process**, which is the process that computer scientists use to create new computer software.

Today, you're going to use parts of the software design process to make your own "unplugged" games without using a computer.

READ ALOUD

Part 1: Tag, You're It!

First, we're going to create our own version of tag called "conditional tag." **conditionals** are instructions to follow if something is true. They are often in the form "if _______, then _______." For example, you might say: "If it is cold outside, then wear a sweater." Conditionals are very important in computer science because they allow programmers to tell computers how to react to various kinds of input.

In this activity, there will be one green and one yellow beach ball corresponding to two different kinds of "its." Tagging will now occur by touching someone with the beach ball instead of with your hand. Because there are only two balls, there can only be two people who are "it" at a time. You'll specify what happens when a tag occurs. For example, you could create a game very similar to freeze tag by instructing the yellow "it" to freeze people, and the green "it" to unfreeze them. What are some other things that could happen when a tag occurs?

(Give kids an opportunity to answer.)

Possible Examples:

- The person tagged becomes "it".
- The tagged person has to lie still on the ground and pretend they are a hot dog.
- The tagged person locks arms with the "it" they were tagged by to create a chain.

We can think of this as a conditional: "If you are tagged by the green ball, then ______. If you are tagged by the yellow ball, then _________. If _____ happens, then the game is over." Working in groups, take 5 minutes to fill out pages 12 and 13 to design your own conditional tag!"

FACILITATOR TIPS

Separate kids into groups of about 5 and give each group no more than 5 minutes to design their tag games.

READ ALOUD

Now partner with another group and take turns playing each other's conditional tag for 5 minutes each. Each group must explain their rules prior to playing.

FACILITATOR TIPS

Make sure everyone's shoes are tied before starting to play tag. If you're in a large area, be sure to define the boundaries of play before you begin.

READ ALOUD

Reflection Questions:

- How did it go? What worked? What didn't work?
- What might you change to make your game better?

By playing each group's game, you performed an important step in the software design process called **user testing**. User testing allows computer scientists to evaluate and improve their projects. Be sure to use the feedback from your custom tag game as we create new games in Parts 2 and 3.

READ ALOUD

Part 2: Finding the Pattern

There are many types of tag, including:

- freeze tag, where you have to stand still when you are tagged until a teammate untags you;
- zombie tag, where one person starts as "it" and everyone that is tagged becomes "it" until there is only one person who isn't "it;"
- battle royale tag, in which everyone starts as "it" and when tagged you are out until there is only one person remaining; and
- the custom version of tag you created in the last activity.

What are some other versions of tag that you have played?

(Give kids time to answer.)

Possible examples:

- Toilet tag
- Bridge tag
- Animal tag
- Shadow tag

Thinking about these different versions, what are some similarities? In other words, what things make a game a version of tag and not a completely different game?

(Give kids time to answer.)

Possible examples:

- At least one person is "it."
- The "it" can tag people.
- Something happens when a tag happens.

What are some differences in these versions of tag?

(Give kids time to answer.)

Possible examples:

- A different number of people can start as "it."
- Different things happen when someone is tagged.
- Some kinds of tag have an end condition (the game ends when everyone is frozen or time runs out) and some don't.

Finding similarities and differences in data, like we just did, is known as **pattern recognition**. As humans, we have a tendency to do certain tasks without reflecting on them much. For example, you probably don't give much thought to how you solve the problem "2+2," you simply answer "4." When computer scientists program a computer to do a task that is second nature to humans, the programmer must reflect on how to solve the problem for several small cases, and then find a pattern that they can program a computer to follow.

Thinking about what worked and what didn't in the user testing of the tag game you created in Part 1, each group will create a new iteration of tag. Be sure to consider the elements of the other versions of tag we just discussed. What are your favorite parts and how might you put these together to create a new game that others would enjoy playing? Use this information and take 5 minutes to design an entirely new version of tag using the worksheet on pages 12 and 13!

FACILITATOR TIPS

These versions of tag have no constraints related to conditionals or the use of beach balls. Encourage youth to be really creative in designing a game that is safe and fun to play in the space provided.

READ ALOUD

Now, pair up with another group and play each other's versions of tag for 5 minutes!

Reflection Questions:

- How did it go? What worked? What didn't work?
- What might you change to make your game better?

Take It Further Discussion (optional)

One thing you may have noticed is that many versions of tag end in different ways. Some versions, like the original, just go on for a set amount of time. Others end when an event happens, such as when only one person remains. When a computer scientist describes a pattern to a computer, they must ensure that it ends. If not, the computer may never get a result or may crash.

Another thing you might have noticed is that someone always has to start as "it," even though they have never been tagged. In CS we call this a 'base case'; it normally has to be manually specified by either the user or the programmer before the computer can start to follow the pattern.

READ ALOUD

Part 3: Let's Play!

In the previous activities, we looked at various versions of tag, found the patterns among the different versions, and then created our own versions. When we did this, we knew the category of activities we were looking at was "tag" games. This time, we'll look at several typical playground games and try to determine the categories they fit into.

Consider the following games:

- Basketball
- Capture the flag
- Dodgeball
- Duck-Duck-Goose
- Simon Says
- Soccer
- Tag
- Ultimate frisbee
- Volleyball

(Kids can see this list of games in their Youth Workbook.)

Let's use pattern recognition to find things that several of these games have in common.

Possible examples:

- Several involve the use of a ball.
- Several use a type of net.
- Several involve two teams playing against each other.
- Several involve elimination.

READ ALOUD

What are some differences in these games?

Possible examples:

- Some use hands, some use feet, some use both.
- The type of equipment used (flag, frisbee, ball, etc.).
- The dimensions of the field of play.
- How the game is scored.
- Individual vs. team.

Aside from all being activities that can be played on a playground, the items on this list don't belong in one single group like all of the tag games did. Take a moment to group these activities into categories you think are relevant. Try to create these in such a way that each activity only belongs to one category. For each category you create, list all of the playground games that fit into that category. Once you've done this, create a brief description of each of your categories. For example, you might create a category whose description is "games where a team tries to score points" or "games that use a ball."

(Give kids a few minutes to discuss amongst their groups.)

Okay, let's discuss some of the categories you came up with.

Possible examples:

A set with three categories:

- Games that involve scoring and points (basketball, volleyball, soccer, ultimate frisbee).
- Games where players are eliminated (Simon Says, dodgeball).
- Games involving chasing and fleeing (tag, duck-duck-goose, capture the flag).

A different set with two categories:

- Games where teams play against each other (basketball, volleyball, soccer, ultimate frisbee, dodgeball).
- Games where every person is for themself (tag, duck-duck-goose, Simon Says).

Congratulations, you just 'abstracted' concepts from several specific examples. ***Abstraction*** is another part of the software design process; it involves identifying and extracting relevant information to define main ideas.

Abstraction is important in computer science because it allows us to generalize the things we build and to create solutions that can solve multiple similar problems. You can now use one of your generalized concepts to create a custom solution. Working in your same groups, you're going to design your own playground games that fit into one of the categories you created. It must be safe and fun to play in the space we have today, using the supplies provided. Remember that conditionals can be a powerful tool to describe the rules for a game.

Now, take 10 minutes to work in your groups to design a new playground game. Use the blank lines in your workbook to write down the rules.

Let's play!

FACILITATOR TIPS

- To save time, for this part you can have the entire class design a game together, choose to have groups partner up and play each other's activities, have each group present their activity then vote on which one to play as a group, choose an activity at random to play, or any other option you want. Do what fits best with your space, the number of children you are doing this activity with, etc.
- Make sure that the rules of the activity are clearly explained before playing.

READ ALOUD

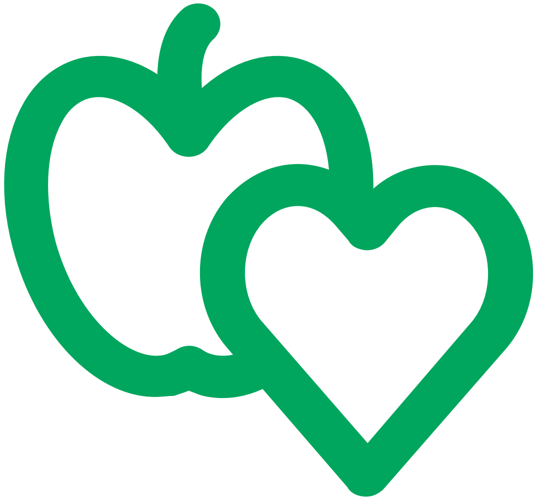
Reflection Questions:

- What did you learn?
- Why is it important?
- How might you use this information next?

READ ALOUD | PILLAR TIE-IN

Take It Further Discussion (optional)

With Program Your Playground we experienced how something fun like physical activity can help us learn about computer science, but another important part of healthy living is good nutrition. There are many ways that the computer science concepts we've just learned can be relevant to healthy eating. Abstraction, for example, is also used in Nutritional Science. When analyzing how different foods affect humans, nutritional scientists often focus on macronutrients instead of focusing on each food individually. There are three macronutrients that make up the foods we eat: carbohydrates, proteins and fats. Most foods contain a mixture of one or more macronutrients, but they are generally classified according to the one they contain most of. This means that instead of looking at pasta, bread, potatoes and other such foods individually, a nutritional scientist can instead just look at the more generalized category of "carbohydrates."



Computer scientists have developed apps that allow people to keep track of the food they eat. These apps monitor how much of each macronutrient is consumed and can provide feedback on what components we are getting too much of, and which we aren't getting enough of. This can help us make smarter food choices.

Educational Standards

NGSS Science and Engineering Practices

Using Mathematics and Computational Thinking:

- Elementary 3-5: Organize simple data sets to reveal patterns that suggest relationships.
- Middle School 6-8: Create algorithms (a series of ordered steps) to solve a problem.

Engineering Design:

- Elementary 3-5: Define a simple design problem reflecting a need or want that includes specified criteria for success and constraints on materials, time or cost.
- Elementary 3-5: Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem.
- Middle School 6-8: Analyze data from tests to determine similarities and differences among several design solutions to identify the best characteristics of each that can be combined into a new solution to better meet the criteria for success.
- Middle School 6-8: Develop a model to generate data for iterative testing and modification of a proposed object, tool or process, such that an optimal design can be achieved.

CSTA Computer Science Standards:

- 1B-AP-09 3-5: Create programs that use variables to store and modify data.
- 1B-AP-10 3-5: Create programs that include sequences, events, loops and conditionals.
- 1B-AP-11 3-5: Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
- 1B-AP-12 3-5: Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
- 2-AP-12 6-8: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- 2-AP-17 6-8: Systematically test and refine programs using a range of test cases.

Teaching Computer Science Beyond NYSD

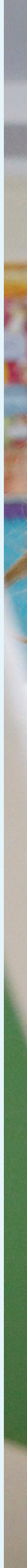
Interested in more great CS activities? There are a lot of excellent resources out there.

Here are some that we recommend:

- ***Code Your World*** is the 2018 NYSD challenge, which also teaches CS concepts through hands-on learning. If you enjoyed *Game Changers*, be sure to check out *Code Your World* at **bit.ly/NYSDcodeyourworld**.
- **4-H CS Playbook** (**www.4-h.org/CSplaybook**) is the ultimate guide for educators with little to no CS experience who are looking to bring CS into their classrooms or clubs in a hands-on, fun and experiential way.
- **CS First** (**g.co/csfirst**) offers an introductory, video-based computer science curriculum that teaches students foundational skills using Scratch. Try another hour-long activity, like Create your own Google logo, or even a full theme, like Storytelling!
- **Code.org** provides curricula for K-12 computer science educators and organizes the annual Hour of Code, a global movement that encourages kids to create with code for one hour. Join the movement and participate in #HourofCode this #CSEdWeek in December.
- **Scratch** (**scratch.mit.edu**) is the world's largest and friendliest creative coding community for kids. Encourage kids to create new projects and explore, and you can explore their Scratch Educator community for teaching resources.

Don't forget to share photos from your NYSD experience on social media using **#4HNYSD**.

Your feedback helps us improve NYSD each year! Please take a few moments to fill out this survey about your NYSD experience: **www.4-h.org/NYSDsurvey**.



4-H
NATIONAL
YOUTH
SCIENCE DAY

JOIN THE NYSD MOVEMENT!

Throughout the month of October, we estimate more than 200,000 kids will take part in NYSD at events across the country. Help us exceed our goal by joining the NYSD movement — together we can make hands-on STEM and CS education accessible to all!

- **PREP:** Get ready to facilitate *Game Changers* by reading through this guide. Focus on the Facilitator Preparation section for a concise overview of how to prepare.
- **PLAN:** Plan your NYSD event (or events) for any time during the month of October. An event can be as simple as taking over a class lesson or teaching a few kids at home, or as big as planning a community event for hundreds. Beyond October, re-use this kit anytime to bring CS to more kids!
- **CHECK IN:** Visit **www.4-H.org/NYSD** for the latest updates on NYSD. As October gets closer, we'll add details and resources to help you make the most of NYSD, including promotional materials and printable resources.
- **SHARE:** Tell your friends and colleagues about NYSD, and don't forget to share on social media using **#4HNYSD**.

Your feedback helps us improve NYSD each year! Once you've completed the *Game Changers* challenge, please take a few moments to fill out this survey about your NYSD experience: **4-H.org/NYSDsurvey**.

GAME CHANGERS

NATIONAL YOUTH SCIENCE DAY

Support from:

Google

Donaldson. FILTRATION SOLUTIONS

HughesNet.

LOCKHEED MARTIN

U.S. AIR FORCE

EXTENSIONSERVICE

USDA United States Department of Agriculture

National Institute of Food and Agriculture

In 4-H, we believe in the power of young people. We see that every child has valuable strengths and real influence to improve the world around us. We are America's largest youth development organization—empowering nearly six million young people across the U.S. with the skills to lead for a lifetime.

Learn more online at: **www.4-H.org/NYSD**