Professional Development Situation: Training
Skill Focus: Developing a STEM Identity
Time Required: 120 minutes

YOU'RE A COMPUTATIONAL THINKER

Participants will learn how they are computational thinkers, how these practices fit into other activities, and how to develop computational thinking practices with youth.

Agenda

Welcome and introduction - 5 minutes
Introduce the skill with "Figuring it Out" – 20 minutes
Learn to play "Game with No Instructions" – 30 minutes
Discuss engaging youth in computational thinking – 20 minutes
Exploring how computational thinking can be used in other areas – 30 minutes
Personal reflection – 10 minutes
Conclusion – 5 minutes

Materials

- Computer with internet connection and speakers
- Paper and pens/pencils (for taking notes and personal reflection if not using a computer)
- Name tents – preprinted (recommended) or blank
- Computational thinking activity (1 set per group)
- One die
- One Computational Thinking Kit
- Pens, pencils (one for each person)
- Flip chart paper and markers
- Handouts

- Defining Computational Thinking (1 per participant)
- Computational Thinking activity from Code.org (1 per participant)
- Adding computational thinking to other areas (1 per group)

Before the Session

- **Read this training guide** to familiarize yourself with the content and to personalize the activities to best suit your style. Watch all videos and read informational materials.
    - *Italics indicate text that can be read aloud or emailed to participants.*
- Send a reminder email about the meeting. Determine if any participants require accommodations (sight; hearing; etc.). If you are not providing computers, remind them to bring their own.
    - *The next professional development opportunity to enhance our STEM skills will be on DATE at TIME at LOCATION. Our focus for this session will be "You're a Computational Thinker". Let me know if you require any accommodations to participate in the training. I am happy to answer any questions you have and look forward to seeing you at the workshop. I can be reached at CONTACT INFO.*
- Gather all materials and make copies of handouts
- Preprint name tents (recommended) or have blank name tents ready.
- Develop a list of possible questions participants might have during the training and your responses to those questions. Review any key terms or ideas that may be unclear.
- Prepare one piece of flip chart paper or a white board with the label, "Problem-Solving Strategies" and one with "Using Computer Science."
- Test the audio and video equipment.
- Write "Ideas for Incorporating Computational Thinking" on a flip chart or whiteboard.

Training Outline

Welcome and Introductions (5 min)

- Greet participants as they arrive. Make sure everyone feels welcome and comfortable. Point out restrooms, refreshments, etc.
- Have participants pick up preprinted or blank name tents and, if needed, write their name and program on them.
- Introduce yourself and the topic: "You're a Computational Thinker".

- If you have 12 or more participants, divide into groups of four and have each person introduce themselves to their group. If you have less than 12, have each person introduce themselves to the whole group in 30 seconds or less.

Introduce the Skill with "Figuring it Out" (20 min)

- You will be using the unplugged "Computational Thinking" activity from code.org.
    - *Unplugged refers to computer science activities that do not require a computer. Computational thinking is a way of thinking in STEM, like the engineering design process, that helps solve problems. Specifically, it is the process of devising solutions that a computer can execute. However, you don't have to use computers to learn about computational thinking practices. Today we will focus on using unplugged activities to teach about computational thinking.*
- Introduce the activity "Figuring it Out" in the "Computational Thinking" lesson plan.
    - *Let's start out by exploring computational thinking with a challenge. The challenge is to sum up all the integers from 1 to 200 in your head.*
    - (Write on a whiteboard or piece of flip chart paper that everyone can see. "Find the sum of numbers from 1 to 200.")
- Pause for a moment.
    - *I'll give you about one minute to work on the challenge. Then you can share your solution.*
- Watch the group. Who seems overwhelmed? Who seems to be working on the problem in their head?
    - *Who is ready to share their solution?* (Pause to see if anyone wants to share.) *Who thought the problem was so hard that they didn't really try? Is it a big problem to do in your head? Let's see if we can break it up into smaller pieces that are easier to manage. We'll start with the two ends. What is the sum of 200 plus 1?* (Pause for someone to say 201.) *What about 199 plus 2?* (Pause) *198 plus 3? Are you seeing a pattern?* (Pause so someone can describe the pattern.) *How many pairs will add up to 201? What is the last pair in this pattern?* (It would be 100+101=201.)
    - *That means we have 100 pairs of numbers that equal 201. Does that make it easier to figure out the sum of all the numbers?* (Pause) *If we added all these pairs together, we would be adding 201 + 201 + 201 + 201 for 100 times Does anyone have a quick way to add the same number 100 times?* (Pause for someone to suggest that the answer is 201*100=20,100.)

- *Now let's think about how we could make this a solution that would work for other challenges, like finding the sum of all numbers between 1 and 2000. What would that look like If we think about it, I think we could even come up with a solution that would work for any set of integers.* (If the group is interested, give them time to think about the algorithm, or just continue with sharing the solution.)

- Write on the whiteboard or piece of flip chart paper that you used before. "The sum of all numbers between 1 and _____ is _____/2 * _____+1" (Pause so the group can discuss this solution if they want to).
- Introduce the vocabulary with participants:
  - *Before I connect our challenge back to computational thinking, I have four new words to introduce. We will use these terms to reflect on how we solved this challenge together. The definitions are on your "Defining Computational Thinking" handout; algorithm, decompose, abstraction and pattern matching.*
- Give participants a minute or two to read through the definitions.
  - *How do you think computational thinking and the vocabulary terms at the bottom of the page are related to the challenge?* (Pause for input from the group. Participants need to understand that these vocabulary words refer to the practices or steps that learners use in computational thinking.)
  - *This was a really big challenge when we started, but we used decomposition to break it down into smaller pieces that were easier to manage.*
  - *Then we noticed the pattern of pairs adding up to 201. Pattern matching made the challenge a little easier.*
  - *In the end, we used abstraction to think about how we could describe a solution that would work for any set of integers, and we wrote that as an algorithm.*

Learn to Play "Game with No Instructions" (30 min)

- Hand out materials for each group (1 die, computational thinking handouts, and a pen or pencil).
  - *Now we are going to play a "Game with No Instructions." With your small group, you will figure out how to play the game using the "Computational Thinking" worksheet in your handout. Read the user experience scripts to get an idea of how others have played the game. Look for the patterns. I'll give you 20 minutes to play the game at your table. Raise your hand if you want more instructions for your group.* (For groups that ask for more instructions, tell them to circle the words that are identical for all

three players, and underline the words that change from player to player, then have them look for the pattern.)

- After 20 minutes, or after everyone has played the game, gather the group together in one area of the room away from where they played the game.
    - *We are going to finish this activity with a Flash Chat.*
- *This is a strategy you can use when you lead computer science activities.* (If you have 12 or more participants, divide into smaller groups to discuss these questions. If you have less than 12, you can discuss as a single group).
    - *If a problem is too hard, what should you try to do?*
    - *What strategies did we use today to solve a problem that was too hard?*
    - *If you have a problem that is just a little different from a problem that you have a solution for, what could you do?*

Discuss Engaging Youth in Computational Thinking (20 min)

- *What would you expect to happen if you led this activity with youth at your program? What would they get out of it? Would they think of themselves as computational thinkers? Share your thoughts in the chat box. (Pause so participants can type.)*
- *When we do computer science in 4-H, it isn't just about the content, it is also an opportunity to support positive youth development. Computer science activities can help youth develop a sense of confidence and competence. Computational thinking builds problem solving and critical thinking. So now we are going to watch a video of youth doing this activity and think about how Dagen facilitates the experience to support developing a STEM identity, - seeing themselves as a computational thinker.*

- Watch the activity overview video, Playing a Game without Rules
    - *What did you notice in the video? Share your observations in the chat box.* (Pause so participants can type.)
- As participants are typing, cue up the skill video.
    - *Now let's watch a second video that focuses on how Dagen facilitates the activity. As you watch the video, notice the positive and negative emotions youth experience. Are they learning to manage their emotions? Notice how Dagen adapts the activity to support youth in seeing themselves as a computational thinker.*
- Watch the skill video, Developing Computational Thinkers. Use the questions below to facilitate a discussion in the chat box about the videos.

- *How does Dagen adapt the activity to support youth in seeing themselves as computational thinkers? Do you think he was successful?*
- *What did he add that wasn't part of the activity when we did it?*
- *What positive and negative emotions do you see in the video? Are they learning to manage emotions in this activity?*
- *How would you adapt the activity to meet the needs of the young people you work with?*
- *Positive youth development needs to be an important component of your thinking about computer science and computational thinking. We want this to be more than developing knowledge, we want to use these activities to help youth develop their full potential as thinkers, problem solvers and innovators.*

Exploring how Computational Thinking can be used in other areas (30 min)

- Distribute the handout "Adding Computational Thinking to Other Areas."
  - *With your small group of 2-4, you will brainstorm about how computational thinking fits into other areas or disciplines. Thinking of computational thinking as a problem-solving method, come up with at least 3 ways of incorporating computational thinking into each area on the handout. Focus on activities that are part of the programs you work in for each area. For example, if youth in your program play basketball, think about integrating computational thinking in that area. If you lead a horse club, think about integrating it in the activities your club does. Focus on activities you already do, rather than coming up with new activities. You have 15 minutes to work with your group, then we will come back together at ____ (Give a specific time.).*
- As groups are working, circulate and answer questions. If groups are really struggling, you can share these examples.
  - Cooking – a recipe is an algorithm used to help food taste the same every time.
  - Team sports – the rules of a game is an algorithm. In games like golf, players practice movement patterns to get consistent results.
  - Physical fitness – athletes use decomposition to figure out how they can increase their fitness, change their diet, or improve their performance.
  - Music – patterns in music help define jazz, country music or R&B.
- As groups begin to finish their discussion, pass out markers and a piece of flip chart paper. Have each group write their three best ideas on the paper.

- Have each group briefly share the ideas on their chart. If you are short on time, post the charts and give participants five minutes to walk around and read the other charts.
- Bring the whole group together and ask:
    - *Why is learning computational thinking important for the youth in your program? How will it help prepare them for their future?*

Personal Reflection (10 min)

- Be sure each person has a computer or paper and pen.
    - *Now that you have learned what computational thinking is and thought about how you could incorporate it in your work, take 10 minutes to write about how you, or the youth you work with, are computational thinkers. Why is that important to you? How does it empower you or challenge you?*
- Give 5-10 minutes for participants to reflect quietly. If time allows, ask if anyone would like to share their reflection.

Conclusion (5 min)

    - *Thank you for coming to the session today. We learned what computational thinking is, how you can incorporate it into activities you already do, and how you are a computational thinker. Following the session, I will share the notes from the charts we created today. You can also take the Computational Thinking activity with you to use in your programs.*
- Answer any final questions participants may have.

After the Session

- From the flip chart paper, compile ideas for incorporating computational thinking into activities already being done.
- Within three weeks of the training, send an email to participants with the message below and the lists they generated in the workshop.
    - *Thank you for your participation in the recent Click2Science training on "You're a Computational Thinker". I hope you found it useful. Consider meeting with a co-worker, supervisor, or friend to share what you learned. I look forward to continuing our learning at the next session on SKILL/FOCUS on DATE at TIME at LOCATION. Please let me know if you have any questions. I can be reached at CONTACT INFO.*

Want to Earn Credit? Click2Science has teamed up with Better Kid Care to provide continuing education units. Check it out at: http://www.click2sciencepd.org/web-lessons/about

This material is based upon work supported by the National Science Foundation under Grant No. 1840947

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Defining Computational Thinking

The International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) have collaborated with leaders from higher education, industry, and K–12 education to develop an operational definition of computational thinking. The operational definition provides a framework and vocabulary for computational thinking that will resonate with all K–12 educators

Computational thinking (CT) is a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data
- Representing data through abstractions such as models and simulations
- Automating solutions through algorithmic thinking (a series of ordered steps)
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem solving process to a wide variety of problems

The definition for computational science listed above can be found at: http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

Decompose describes the thinking practice of breaking a problem down into smaller pieces.
Pattern Matching describes the thinking practice of finding similarities, or patterns, between things.
Abstraction describes the thinking practice of pulling out specific differences to make one solution work for multiple problems.
Algorithm is a list of steps that you can follow to finish a task or reach a solution.

Adding Computational Thinking to Other Activities

*Add three additional areas you have in your program at the bottom of the list and brainstorm ideas for these areas as well.

Area	Activity	How to add Computational Thinking
Health		
Physical Fitness		
Team Sports		
Cooking		
STEM class		