Professional Development Situation: Virtual Training
Skill Focus: Enabling Active STEM Learning
Time Required: 120 minutes

BREAK IT DOWN

Participants will learn how breaking problems into smaller, manageable parts is key to computational thinking and build this skill.

Agenda

Welcome, introductions and explanation of Zoom – 5 minutes
Introduction to vocabulary – 5 minutes
Practicing computational thinking – 25 minutes
Process the experience– 10 minutes
See the skill in action – 10 minutes
Using a graphic organizer – 10 minutes
Engaging in active learning at the computers – 40 minutes
Process the experience – 10 minutes
Conclusion – 5 minutes

Materials

- Computer with internet connection, camera and speakers
- Paper and pens/pencils (for taking notes)
- Monster activity supplies (1 set per person)
    - Monster catalog
    - 3 blank monster faces
    - Markers
    - Scissors
- Step-by-step activity
    - Headphones are recommended

- Handouts to send as attachments in pre-session email and download onto your computer for screensharing
    - Defining Computational Thinking
    - Graphic Organizer
    - Monster Catalog – pages 7-23 of Lesson 3: Computational Thinking Lesson Plan (click on open lesson plan to download)

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.). You can set up a meeting and send an email about the meeting through Zoom. The following should be included with the Zoom invite you send.
    - *The next professional development opportunity to enhance our STEM skills will be on DATE at TIME. Our focus for this session will be "Break it down". This is a virtual training, so before we begin the session, gather the materials and print one copy of each of the attached handouts. You will also need at least 3 pieces of blank paper.*

      Please have paper and a pen or pencil for taking notes.

      You will also need markers, a pair of scissors and a computer with an internet connection, a camera and speakers or headphones.

      Please try to avoid calling in on a phone. It will be more beneficial to use a computer with video and the chat box is more difficult to use with a mobile phone.

      There are three attachments to print. You should not fill them out before the session—we will do that during the training.
        - Defining Computational Thinking
        - Graphic Organizer
        - Monster Catalog – pages 7-23 of Lesson 3: Computational Thinking Lesson Plan (click on open lesson plan to download)

Finally, you should have Zoom (or another video conferencing tool) installed on your computer before the session. You should test the tool before the meeting. If you are using Zoom, there are tutorials for using the video, audio and screen sharing features. These can be found in the lower left corner of the screen after you have logged into zoom.

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 download handouts before the session to use as visual aids if necessary.
- Test the audio and video equipment.

Preparing for Virtual Trainings

- This training is written for Zoom, but you can use any virtual meeting platform you have access to and are familiar with. Practice using the virtual meeting software before the meeting and be sure you know how to use all the features that will be used in this training.
- Participants can download, install, and use the free version of Zoom. They will be able to use the whiteboard feature and breakout rooms as long as the host has the Pro version. As host, you need the Pro version to schedule a session that is longer than 40 minutes. There is a cost for the Pro version. You can choose from one of the plans listed on their website or you can contact Zoom to customize a plan for your circumstances.
    - Zoom plans and pricing https://zoom.us/pricing
    - Zoom for Education https://zoom.us/education
- If your organization has Zoom, they might have a URL that you should use to sign up for Zoom. If you are using a different platform, provide participants with the URL they need to download the platform. Set up Zoom (or whatever system you are using) before sending the email reminder because there is a calendar feature in Zoom so you can add your training session to participants' calendars.
- We recommend at least two facilitators for a virtual CSPD training. A co-facilitator will help you manage participants and answer questions in the chat box, but they will not have all the abilities in Zoom that a host has.
- Zoom allows up to 50 people to use video in a meeting with the Pro version. You can schedule two sessions if you have over 50 people. When participants join the meeting, their name should appear which is helpful for calling on someone.

- We recommend all facilitators have dual monitors so you can share one screen and have the participants and breakout rooms on the other screen. It will make it easier for you to manage everything.
- For this training, you will need to know how to share your screen, use the chat, breakout rooms, and whiteboard. You may also want to record your meeting. You must enable breakout rooms before the session begins.
    - Learn how to use the chat feature. Be sure you know how to save the chat. You must do this before you end the meeting.
    - Learn how to use breakout rooms. There are tutorials on how to set up and manage breakout rooms. You need to enable breakout rooms before you start the meeting. We recommend assigning participants automatically unless you have groups within your training that you want to collaborate together. Be sure you know how to set up breakout rooms, assign participants, manage the rooms and broadcast a message to all rooms.
    - Learn how to use the whiteboard feature. Be sure you know how to insert text, draw on, save and clear your whiteboard.
    - Learn best practices for meetings and webinars that explain how to engage participants and follow up after the session.

Training Outline

Welcome, introductions and explanation of Zoom (5 min)

- Greet participants as they join the meeting and ask them to introduce themselves in the chat by sharing their name and where they are from. Encourage everyone to turn on their video. You may want to record the training, especially if someone is absent.
- Introduce yourself and the topic for this training, “Break it down”.
    - *Today, we may be using Zoom (or another platform you’ve selected) in ways you haven’t experienced before. We are going to* [what you say will depend on the size of your session, options are listed below]
- If you have less than 12 participants, have each person introduce themselves and share one expectation or hope for today’s workshop as you say their name. This will assist people in knowing when it is their turn to speak.
- If you have 12 or more participants, explain how breakout rooms work. Explain that everyone will introduce themselves and share one expectation or hope for today’s

workshop in their breakout room. Tell them you will bring them back from their breakout room in 8 minutes.

- After 6 minutes, use the "Broadcast a message to all" button and tell the participants they have 2 minutes left to finish introductions.
- After 7 minutes, click the "Close All Rooms" button. This will give each breakout room a 60-second countdown, and will automatically end the breakout session and return them to the main room when the timer ends.
- Direct participants to https://support.zoom.us/hc/en-us/articles/115005769646-Participating-in-Breakout-Rooms if they need help with the breakout rooms.

Introduction to vocabulary (5 min)

- *Today we will focus on strategies that will actively engage young people in computer science. We're going to start with an unplugged activity from Code.org to practice. Unplugged activities are fun computer science experiences that reinforce important skills but do not require a computer. I'm going to introduce this unplugged activity with a video.*

- Share your screen and watch the video, "Computational Thinking". https://studio.code.org/s/20-hour/stage/3/puzzle/1 (1:34 minutes)
  - *There was some vocabulary in the video that may be new to you. She mentioned four steps of computational thinking. We will be using these terms today, so I want to be sure we are all thinking about them in the same way. You should have printed the "Defining Computational Thinking" handout, you can refer to this handout the rest of the session to help you learn the vocabulary.* If your group is not familiar with computational thinking, use these steps to introduce the vocabulary with participants. If they are familiar with the term, ask someone to summarize what computational thinking is and why it is important for young people.
  - *ISTE and CSTA define computational thinking as "a problem-solving process that includes formulating problems so that we can use a computer or other tools to help solve them. It includes logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking and generalizing the solution to a wide variety of problems." http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf*
  - *The four steps, or practices of computational thinking were explained in the video and they are also on your handout. Do you want us to revisit any of these terms or how they are related to this activity?* (Pause for input from the group. Participants

need to understand that the vocabulary words refer to the practices or steps that learners use in computational thinking.)

- Your participants will **decompose** the problem as they identify the different parts of the task and make a game plan.
- Then they will look for **patterns** that are shared by the monsters in the catalog.
- **Abstractions** will help them focus on what parts of the pattern are true for all the monsters – like they all have eyes – and what differs from one monster to the next – like the type of eyes they have.
- Then they should be ready to write a simple set of instructions, or an **algorithm**, that another group can follow to create the monster they've described.

Practicing Computational Thinking (25 min)

- Prepare for the code.org activity Lesson 3: Computational Thinking (click on open lesson plan to download). Divide participants into groups of four. Tell each person they should have one monster catalog, three blank pieces of paper, markers, and a pair of scissors in front of them to do this activity.
    - *We have been asked to help identify monsters found on the planet Zuron. Our computers need to be able to generate images of the monsters based on eye-witness accounts. There are quite a few monsters to describe, and it may seem challenging, but luckily, we have some tools that will help us.*
    - *We are going to use computational thinking to make this task easier. The four practices of computational thinking can be used to make difficult problems easier to solve.*
- *First, we will decompose the problem—we're not just talking about zombies here. Instead, we're talking about breaking a big, bad problem down into something much simpler. Often, big problems are just lots of little problems stuck together.*
- *Then we will look for Patterns—sometimes, when a problem has lots of little pieces, you will notice that the pieces have something in common. If they don't, then they may at least have some striking similarities to some pieces of another problem that has been solved before. If you can spot these patterns, understanding your pieces gets much easier.*
- ***Once you recognize a pattern, you can "abstract out"** (or ignore) the details that make things different and use the general framework to find a solution that works for more than one problem.*
- ***When your solution is complete, you can write it up as an algorithm**, or instructions that allows it to be processed step by step, so that the results are easy to achieve.*

- *Now I'm going to give your groups 20 minutes to work through this activity. You should each come up with an algorithm by yourself. As soon as everyone in your group has their algorithm done have each person read their algorithm to the group and everyone else can draw that algorithm on one of their blank pieces of paper. Then hold up the picture you drew in front of your camera and see if the algorithm produced the correct monster. As you work through the process, create a list of the tasks you do and type them in a word document.*

- Let participants struggle if necessary. After about 15 minutes, or as participants finish, remind them to test their algorithms with other group members.

Process the Experience (10 min)

- *Now I want you to take five minutes in your breakout rooms and discuss the list of tasks you each developed. Cut and paste your list into the chat box or on the whiteboard. What are the patterns? Focus on what is similar and what is different. Then have one person from each group ready to discuss them. (Allow 5 minutes for groups to discuss).*
- *Let's talk about what you noticed about the lists.*
  - *What were the tasks that appeared on all the lists?*
  - *What were the tasks that were only on one or two lists?*
  - *What tasks made this an active learning process?*
  - *Did some lists have the tasks listed in a different order?*
  - *Were these differences important? Why or why not?*
- *The activity you just did focused on components within each of these parts of computational thinking which computer scientists call aspects. However, it is important to note that what we just learned are not the only components of these different aspects. For example, abstraction also includes removing non-essential or non-relevant information in relation to the task you are trying to do. The components you have learned today are the beginning to understanding the broader aspects of computer science.*

See the Skill in Action (10 min)

- *I am going to share a video of a group of youth doing the monster activity we just did.*

- Watch the activity overview video, Computational thinking with monsters

- *What did you notice in the video? Share your observations in the chat box.* (Pause so participants can type.)
- *Now let's watch a second video that focuses on active learning. As you watch the video, notice what the facilitators do to make the activity successful. Pay attention to how they talk about computational thinking.*

- Watch the skill video, Breaking down active learning
- Use these questions to facilitate a discussion in the chat box about the videos.
  - *How does Dagen engage youth in active STEM learning?*
  - *How did the questions Dagen ask youth help facilitate active STEM learning?*
  - *How do you know if youth are engaged in active STEM learning?*
- You can also discuss the questions as a group.

Using a Graphic Organizer (10 min)

- *For this next activity, we're going to work in groups of three or four in the breakout rooms. You will use the graphic organizer you printed out before the session to reflect on two things about teaching computational thinking. 1) How can you keep it an active learning experience and 2) how can youth apply these thinking practices to their real world? I'll give you until ____ (time in 10 minutes) to work in your group. (Put participants back into their breakout rooms).*

- If time allows, go through each section of the graphic organizer and invite groups to share one idea for each practice.

Engaging in Active Learning at the Computers (40 min)

- *Now on our computers we are going to try out a curriculum from Harvard Graduate School of Education called "Creative Computing" which uses Scratch to teach a variety of concepts.*
- *Scratch is a visual programming language designed by MIT. If there is anyone that hasn't used Scratch before please type that in the chat box to me only and I will put you in a breakout room and we can talk through some of the features of Scratch. [This would be easier to do with a co-host so one host can be in the breakout room and the other can be in the main room answering questions].*
- *Get your handout "Debug it!" that you printed before the session.*

  - *The Creative Computing curriculum has six units that you can go through sequentially, but today we're going to jump into unit one to an activity called Debug It! Who can explain what debugging means?* (Pause for response from the group.)
  - *Debugging is figuring out what's wrong in a computer program—it is an important skill to learn in computer science. These practices we've been using today are helpful in debugging, so think about decomposition, patterns or abstractions that can help you figure out what the problem is.*
- Have participants go to http://scratch.mit.edu/studios/475483. This activity is on page 35 of the Creative Computing curriculum.
  - *There are five projects in this studio. We'll do the first one together* [share your screen]*, and then you can work through as many of the others as you have time for. Click on Debug It 1.1. There are instructions on the right side of the screen.*
  - *Read the instructions, then click on the green flag to see what Scratch and Gobo do. Click on the red stop sign to end the program.*
  - *Click on the "See Inside" button at the top right-hand side of the screen and look at the code that makes each sprite move. First look at Scratch's code because you know that it is running correctly.*
  - *What do you see?* (e.g., different colors and words, different shaped boxes that are different colors, or different blocks that fit together).
  - *When you understand the basics of this code, click on the Gobo sprite button at the bottom of the screen to look at the code for this sprite. Try to figure out what is wrong with the code that is causing Gobo to work incorrectly.*
  - *When you think you know what is wrong with the code, make the change and see if it works. Then move on to the next project.*
- If you use a breakout room for participants who have never used Scratch before, say: *On the left side of the screen there is a tab at the top that says "Code", and in that tab are several different colors on the left side of the screen. These tell you how that code will affect your sprite. For example, blue is motion, so when you use motion code, it will cause your sprite to move. If you want your sprite to make a sound, you would use magenta code. There are Scratch tutorials located in the blue navigation bar at the top.*
- If participants have questions have them type them in the chat box, but try not to give them answers, instead ask questions to help them think about solutions. Use the terms decomposition, patterns, abstraction and algorithm as appropriate. Depending on the size of your group, it might work better to put the participants who have used Scratch before into breakout rooms of 3 to 4 and let them help each other.

Process the Experience (10 min)

- Bring the group back together.
    - *As we gather to process our experience debugging, I want to point out a couple of things that you will find helpful as you are facilitating computer science activities. First, moving away from the computers makes it much easier for me to get everyone's attention for this discussion. We obviously can't do this in a virtual session, but you can in a face-to-face setting. Second, as we did this activity, I tried to model active learning strategies that you can use, so let's think about how that worked as we process the experience. I want you to start thinking about how the experience made you feel because emotions are very important in preparing students to learn.*
- Use these questions to guide the discussion and process this experience.
    - *How did you feel about debugging?*
    - *What made you feel frustrated? How did you manage that?*
    - *How did you feel when you figured out what was wrong with the code?*
    - *How would learning how to debug a program help the young people you work with?*
    - *How did you use your understanding of decomposition, patterns, abstraction or algorithms as you were debugging the programs?*
    - *How did understanding computational thinking help you debug?*

Conclusion (5 min)

- *Thank you for coming to the session today. We learned about decomposition, breaking problems into smaller, more manageable parts and saw how this practice is key to computational thinking. We also focused on strategies that you can use to actively engage young people in computer science. Remember to integrate the ideas you put on your graphic organizer when you get back to your program. Following the session, I will share links to the monsters activity and the Creative Computing curriculum so you can use them to practice the skills you developed today.*
- Answer any final questions participants may have.

After the Session

- **Before you end the session,** save the chat box and the whiteboard if it was used. Within three weeks of the training, send an email to participants that includes links to the activities

used in the workshop and any ideas from the chat boxes or whiteboards that might help them in their work with youth.

- *Thank you for your participation in the recent Click2Science training "Break it down". 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 INFORMATION.

  Computational thinking activity link: https://studio.code.org/s/20-hour/stage/3/puzzle/1
  Creative Computing link: http://scratched.gse.harvard.edu/guide/index.html

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.

Graphic Organizer

Decomposition Breaking down a large problem into smaller pieces	Pattern Finding a theme that repeats several times
Abstraction Removing details from a solution so that it will work for a lot of different problems	Algorithm A list of steps used to complete a task