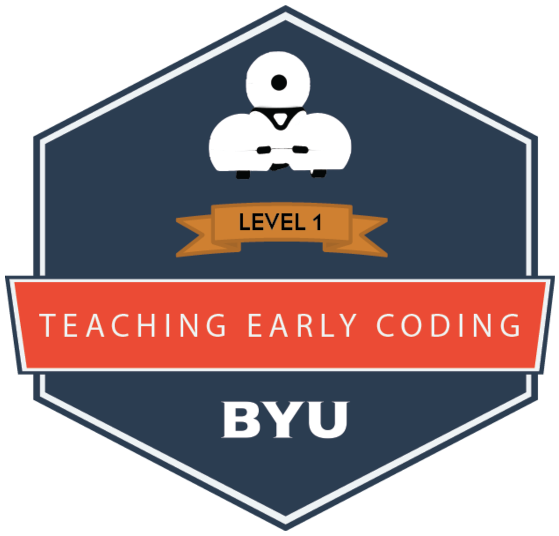


Teaching Early Coding - Level 1

Badge Guide

Enoch Hunsaker
Brigham Young University
2018

Table of Contents

Badge Introduction

Coding is an increasingly important subject to consider teaching to young learners in the 21st Century. In fact, many consider coding to be a literacy of the 21st Century that needs to be learned just like reading and writing in a native language (Code.org, 2013).

In the process of earning this badge, you will learn the following coding concepts:

- Commands
- Loops
- Nested Loops
- Events
- Conditionals

You will also learn how to find resources and create activity plans to teach these coding concepts in your classroom and integrate them with other academic subjects.

It is estimated that this badge should take you approximately **3 hours** to complete.

Recommended Prior Knowledge

Some requirements of this badge will draw on the assumption that you understand the fundamentals of computational thinking (CT). If you aren't familiar with CT, you may consider first earning BYU's Understanding Computational Thinking badge.

International Standards Alignment

ISTE Standards for Educators

- **Learner Standard 1a:** Set professional learning goals to explore and apply pedagogical approaches made possible by technology and reflect on their effectiveness.
- **Facilitator Standard 6c:** Create learning opportunities that challenge students to use a design process and computational thinking to innovate and solve problems.

ISTE Standards for Students

In particular, this badge helps meet the ISTE Facilitator Standard (Standard 6) by empowering teachers to facilitate the following standards for students:

- **Empowered Learner Standard 1d:** Students understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.
- **Computational Thinking Standard 5:** Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions.

Intended Learning Outcomes

1. Articulate rationale for teaching coding to your students.
2. Apply basic coding concepts in a variety of contexts.
3. Facilitate student learning of coding concepts in effective and developmentally-appropriate ways.
4. Evaluate the effectiveness of classroom activities in which coding concepts are taught.

Requirement 1.1: Coding Concepts

Choose a block-based programming platform (e.g., Scratch, code.org) and build one or more block-based code sequences to show that you can apply ALL of the following coding principles (you may build these sequences either as part of an activity, or as a stand-alone project):

- *Commands*
- *Loops*
- *Nested loops*
- *Events*
- *Conditionals*

Take one or more screenshots of your sequence(s) and include them on your submission form along with annotations that describe which coding principles they demonstrate.

BYU's Teaching Early Coding badge levels (level 1 and level 2) were designed on the sequencing found in Code.org's CS Fundamentals Express course. The most direct way to learn the five coding concepts necessary for this badge is to browse through and complete several activities in the first half (lessons 1-18) of that course. You need not necessarily complete every single item within these lessons. Rather, work at a concept until you grasp it, then take a screenshot, add it to your submission form, and annotate it. One screenshot may include multiple or even all of the coding concepts as desired. If you are a little more familiar with the concepts, or if you prefer, you may also choose to use Scratch, a block-based programming language developed by MIT that is similar to Code.org, but which will not provide you with as much guidance throughout the process.

You may want to reference the information below before, during, or after your block-based coding practice to solidify your understanding of these concepts.

Commands

Commands are the basic actions that are already built in to a coding language. They are so fundamental to coding that you won't be able to do anything without them. In the Code.org programming language, you will see the very basic command, "move forward" in Lesson 2.2, the first time you're introduced to any kind of game. In the Scratch create center, you will see all sorts of commands available to you in the Motion, Looks, Sound, and Pen tabs. These commands include things like, “move __ steps” and “turn __ degrees.”

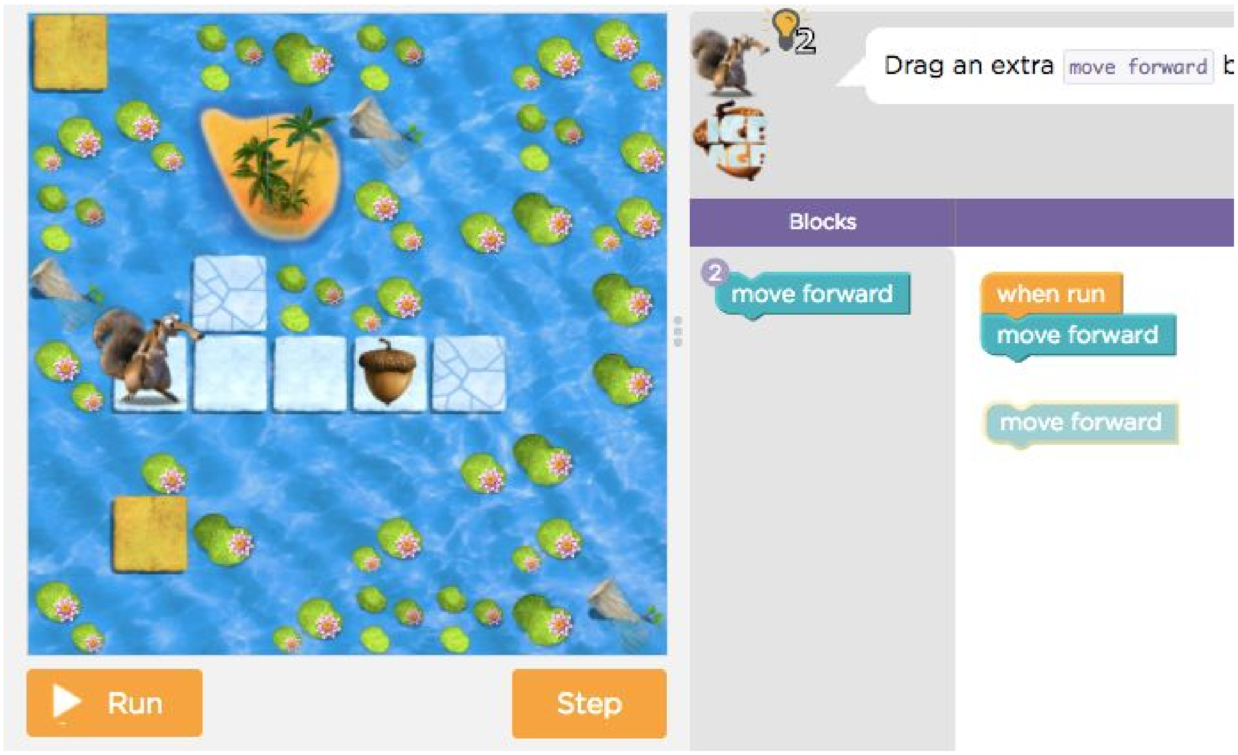
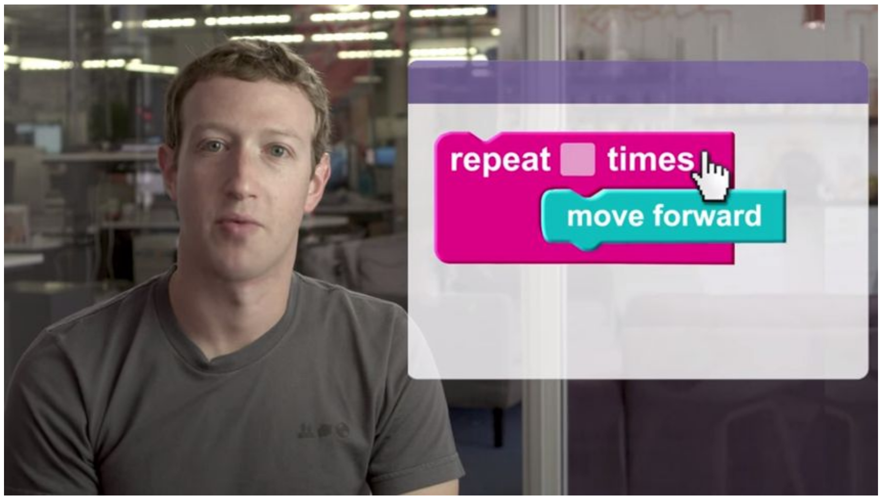


Image 1. Commands in code.org’s CS Fundamentals Express course, Lesson 2.2

Loops

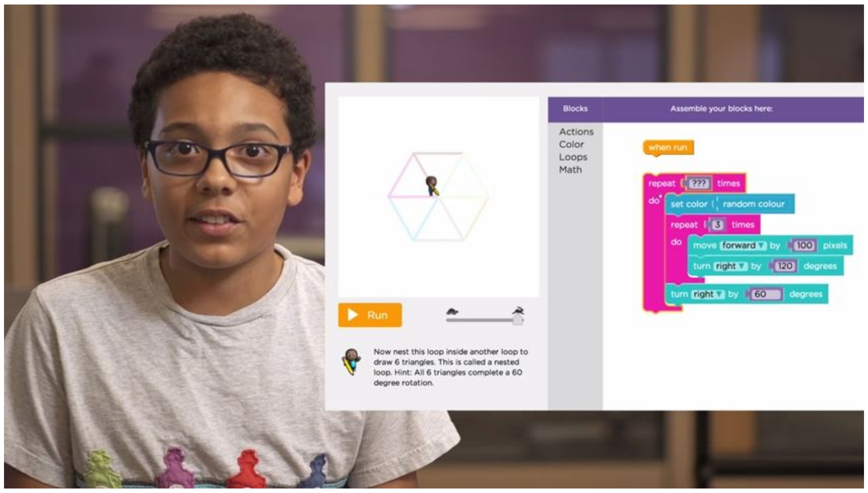
A loop is a programming structure that repeats a command (or set of commands) a specified number of times or until a certain condition is met.



Video 1. Mark Zuckerberg (founder of Facebook) explains loops. Click, copy, or type this link into your browser to view: https://youtu.be/mgooqyWMTxk. ***Note:*** *This video is copyrighted by Code.org under a Standard YouTube License. It is therefore not included in the CC-BY license of the rest of this document.*

Nested Loops

Nested loops are loops within loops.



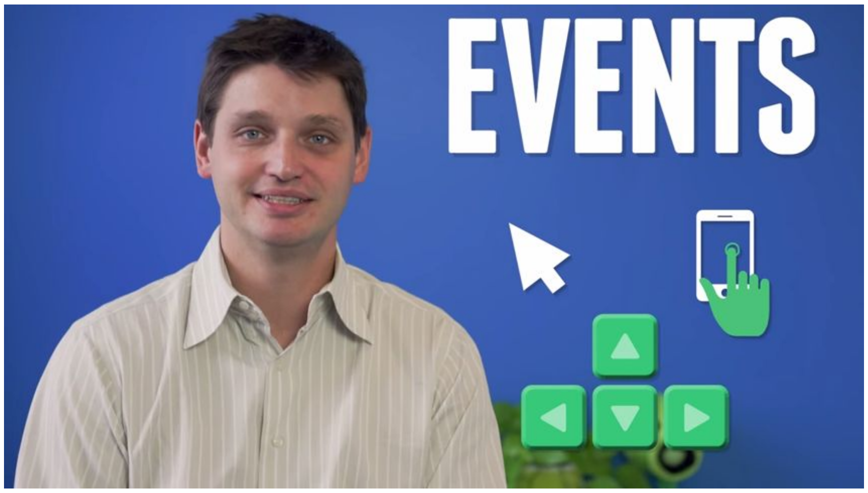
```
when run
repeat ??? times
do  set color random colour
    repeat 3 times
    do  move forward by 100 pixels
        turn right by 120 degrees
    turn right by 60 degrees
```

Now nest this loop inside another loop to draw 6 triangles. This is called a nested loop. Hint: All 6 triangles complete a 60 degree rotation.

Video 2. What nested loops look like in Code.org. Click, copy, or type this link into your browser to view: https://youtu.be/2C0pdRhIZhk. ***Note:*** *This video by Code.org is released into the Creative Commons under a CC-BY license.*

Events

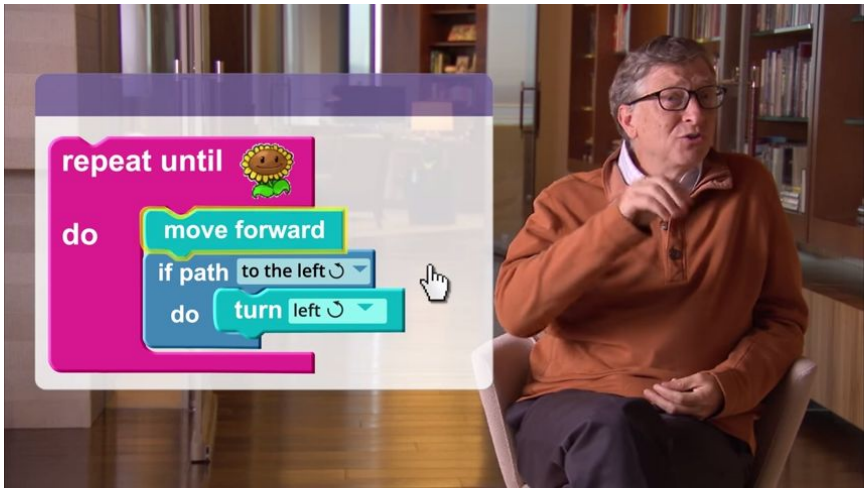
An event tells your program to detect when something happens and take an action when it does. Events are usually external (like a user click or keyboard stroke), but can also be internal to the program, such as when one character in a game intersects with another.



Video 3. A game designer explains events. Click, copy, or type this link into your browser to view: https://youtu.be/7o6ABVPc2Wg. ***Note:*** *This video is copyrighted by Code.org under a Standard YouTube License. It is therefore not included in the CC-BY license of the rest of this document.*

Conditionals

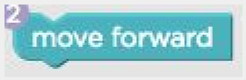
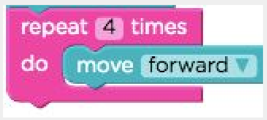
A conditional is a statement that tells a program to execute commands on a condition. It generally follows the form, “If (condition), then (action).”

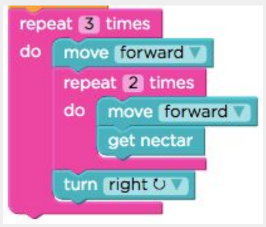


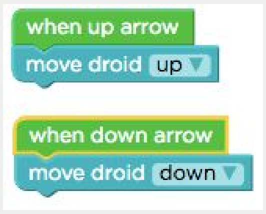
Video 4. Bill Gates (founder of Microsoft) explains if statements (conditionals). Click, copy, or type this link into your browser to view: https://youtu.be/m2Ux2PnJe6E. ***Note:*** *This video is copyrighted by Code.org under a Standard YouTube License. It is therefore not included in the CC-BY license of the rest of this document.*

Comparison Table

The following table may be useful to you as you consider each of the concepts introduced in this badge.

Coding Concepts				
Concept & Definition	**Example(s)**	**Block-Based**	**Javascript**	**Output**
Commands are the basic actions that are already built in to a coding language.	• Move forward • Turn left		`moveForward();`	The specified object moves forward.
Loop. A structure that repeats commands a specified number of times/until a condition is met.	• Move forward 4 times.		`for (var count = 0; count < 4; count++) { moveForward(60);}`	The specified object moves forward 4 times.

Nested Loop. A loop within a loop.	Repeat the following 3 times:
	● Move forward
	● Repeat twice:
	○ Move forward
	○ Get nectar
	● Turn right
		
		repeat 3 times
		do move forward
		repeat 2 times
		do move forward
		get nectar
		turn right
			`for (var count2 = 0; count2 < 3; count2++) {`
			`moveForward();`
			`for (var count = 0; count < 2; count++) {`
			`moveForward();`
			`getNectar();}`
			`turnRight();`
			`}`
				An object (in this case a bee) moves in a pattern to get to and collect nectar.

Event. An action or condition (usually external) that triggers a command or set of commands in the program.	When the user hits the up arrow, move the character up; when the user hits the down arrow, move the arrow down.	
		when up arrow
		move droid up
		when down arrow
		move droid down
			`droid.onkeyup {`
			`moveUp();`
			
			`droid.onkeydown {`
			`moveDown();`
				The droid will move up when the up arrow is pressed and down when the down arrow is pressed.

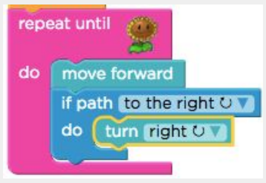
Conditional. A statement that tells a program to execute commands on a condition.	If there is a path to the right, turn right.	
		repeat until
		do move forward
		if path to the right
		do turn right
			`while (notFinished()) {`
			`moveForward();`
			`if (isPathRight()) {`
			`turnRight();}}`
				The character will move forward and always turn right if there is a right turn, until it is done with the game.

Table 1. Coding Concepts Comparison

Requirement 1.2: Learning Reflection

On your submission form, respond thoroughly to the following prompt:

List at least three (3) computational thinking (CT) skills, attitudes, or approaches that you feel were fostered by the activities you completed in Requirements 1.1. (If you don't know what computational thinking is, consider earning the Understanding Computational Thinking badge!)

Whether or not you have completed the Understanding Computational Thinking badge, you may find this table useful for reference. It outlines the basic computational thinking (CT) skills, attitudes, and approaches upon which you are expected to reflect in this requirement.

Components of Computational Thinking		
Skills	**Attitudes**	**Approaches**
• ***Decomposition***: Breaking down data, processes, or problems into smaller, manageable parts. • ***Pattern Recognition:*** Observing patterns, trends, and regularities in data. • ***Abstraction:*** Creating a visual model or simulation of a problem that incorporates only the most important details. • ***Algorithm Design:*** Developing the step by step instructions for solving this and similar problems. • ***Evaluation:*** Ensuring that your solution is a good one.	• ***Confident:*** Believing in one's own ability to solve problems. • ***Communicative:*** Willing and able to communicate effectively with others. • ***Flexible:*** Able to deal with change and open-ended problems.	• ***Tinkering:*** Experimenting and playing. • ***Creating:*** Designing and making. • ***Debugging:*** Finding and fixing errors. • ***Persevering:*** Keeping going. • **Collaborating:** Working together.

Table 2. Components of Computational Thinking

Requirement 2.1: Unplugged Activity

Identify (or create) an unplugged activity that could effectively teach two or more of the coding concepts listed in requirement 1.1 (i.e., commands, loops, nested loops, events, conditionals) to students within your teaching domain. This should be a different activity than you experienced in requirement 1.1. On your submission form, include the following:

- *A link to (or description of) the activity*
- *An indication of which coding concepts it teaches*
- *A written justification of why this activity is appropriate within your teaching domain.*

Terminology

Unplugged Activity: An activity in which students use non-digital physical objects (e.g., a pencil and paper, games, manipulatives, or their bodies) to learn coding concepts.

Teaching Domain: The subject area, grade level, and student demographics of your classroom.

Unplugged Resources

A wide variety of instructional resources for unplugged activities are available for free online. These resources may include lesson plans, printable worksheets, cutouts, etc., and sometimes even videos that depict an example lesson. Among the sites that make these resources available are the following:

- CS Unplugged (https://csunplugged.org)
- CS Fundamentals Unplugged on Code.org (https://code.org/curriculum/unplugged)

You can also find many resources simply by performing a Google search for "unplugged coding activities."

In addition to these online resources, there are a number of unplugged games, interactive books, and manipulative sets that can help you transform your classroom learning experience. Some of these include the following:

- Let's Go Code Activity Set by Learning Resources (LER2835)
- On the Brink game by ThinkFun (52123651)
- My First Coding Book: Packed with ??? and lots more to help you code without a computer! by Kiki Prottsman
- Rover Control game by ThinkFun (EZ112822)
- CodeMaster Programming Logic Game by ThinkFun
- Future Coders Cube Stackers by Alex (890110)

Completing the Requirement

You may complete this requirement either by linking to an existing coding activity, describing an existing activity you find within instructions for a game or coding manipulative set, or even creating your own unplugged activity. Whichever option you choose, you must justify two criteria:

1. The activity teaches a coding concept from this badge (i.e., commands, loops, nested loops, events, conditionals).
2. The activity is developmentally-appropriate within your teaching domain.

The first criterion is straightforward. Simply state the coding concepts the activity teaches.

The second criterion is a little more subjective and will therefore require sound argument and evidence in your written justification. To help you determine whether an activity is developmentally-appropriate within your teaching domain, use what you already know about child development and pedagogical best practices ? to provide a few reasons why you believe the activity would be a good one. Among others, you might consider the following factors:

- **Physical Development**: Do the students have the motor skills they need to complete the activity successfully?
- **Cognitive Development.** Does the activity present the material in a way that bridges their existing knowledge with the new knowledge?
- **Affective Filter.** Will the students enjoy doing the activity?
- **Best Practices.** Does this activity align with best practices for teaching in your target domain?

Requirement 2.2: Codable Robot Learning Activity Evaluation

Watch a video, read an article/blog post, or observe a live lesson that depicts, documents, and/or demonstrates a learning activity with a codable robot of your choice, preferably within your teaching domain or a teaching domain that is closely related to your own. To the best of your ability, evaluate the activity based on the following criteria:

- *The activity and its execution are developmentally-appropriate within the teaching domain.*
- *The activity helps the students to learn one or more coding concepts (any concept(s) will do).*
- *The activity helps the students to learn computational thinking concepts.*
- *The activity helps the students to make connections between the coding and/or computational thinking concepts they learn and other academic areas (e.g., science, math, engineering, language arts, social studies, etc.).*

As part of your evaluation, write a few sentences about whether/how you would improve or adapt the activity to better meet the criteria listed above; indicate where on the PIC-RAT scale you feel the activity falls; and be sure to include some information about which learning activity you evaluated. For your convenience, a form with all necessary fields is already included on your submission form.

Use the evaluation rubric already provided within your submission form to complete this activity. As needed, reference the materials below to complete your assessment.

Understanding the Evaluation Criteria

What is Developmentally-Appropriate?

There is a great deal of conversation about what is "developmentally-appropriate" technology integration in Early Childhood Education (PreK-2nd grade). A prime example is the 2012 joint position statement from the National Association for the Education of Young Children (NAEYC) and the Fred Rogers Center for Early Learning and Children's Media. The statement, Technology and Interactive Media as Tools in Early Childhood Programs Serving Children from Birth through Age 8, outlines several issues, guiding principles, and recommendations for early childhood educators seeking to appropriately integrate educational technology into their classrooms. *If you are an early childhood* *educator, please read the above article and use it to inform your discussion of the developmental appropriateness of the lesson you evaluate.*

There is less discussion and fewer guidelines about what constitutes *developmentally-appropriate* tech integration in secondary or even upper elementary grade levels. However, many of the principles discussed in Early Education literature may be modified to apply to these situations. *If you are an upper elementary or secondary educator, please review the guidelines and questions below and use your "professional judgment" (NAEYC and Fred Rogers, 2012) to determine how they might be adapted for use within your teaching domain.*

Developmentally-Appropriate Guidelines (from NAEYC and Fred Rogers, 2012).

1. "Above all, the use of technology tools and interactive media should not harm children."
2. "Developmentally appropriate practices must guide decisions about whether and when to integrate technology and interactive media into early childhood programs."
3. "Professional judgment is required to determine if and when a specific use of technology or media is age appropriate, individually appropriate, and culturally and linguistically appropriate."
4. "Developmentally appropriate teaching practices must always guide the selection of any classroom materials, including technology and interactive media."
5. "Appropriate use of technology and media depends on the age, developmental level, needs, interests, linguistic background, and abilities of each child."
6. "Effective uses of technology and media are active, hands-on, engaging, and empowering; give the child control; provide adaptive scaffolds to ease the accomplishment of tasks; and are used as one of many options to support children's learning."
7. "When used appropriately, technology and media can enhance children's cognitive and social abilities."
8. "Interactions with technology and media should be playful and support creativity, exploration, pretend play, active play, and outdoor activities."
9. "Technology tools can help educators make and strengthen home–school connections."

10. “Technology and media can enhance early childhood practice when integrated into the environment, curriculum, and daily routines.”
11. “Assistive technology must be available as needed to provide equitable access for children with special needs.”
12. “Technology tools can be effective for dual language learners by providing access to a family’s home language and culture while supporting English language learning.”
13. “Digital literacy is essential to guiding early childhood educators and parents in the selection, use, integration, and evaluation of technology and interactive media.”
14. “Digital citizenship is an important part of digital literacy for young children.”
15. “Early childhood educators need training, professional development opportunities, and examples of successful practice to develop the technology and media knowledge, skills, and experience needed to meet the expectations set forth in this statement.”

Checklist Questions for Developmentally-Appropriate Technology (from Hirschy, 2015).

1. “Will using it influence more than one area of development (social, emotional, cognitive, or physical)?”
2. “Does it allow for varying levels of understanding and abilities in children?”
3. “Will its use build upon children’s previous knowledge and enhance future growth and development?”
4. “Does the technology enhance other activities in the classroom and allow you to build on it to create greater understanding and growth?”
5. “Can it encourage active and creative play?”
6. “Will it allow children to use it together in social contexts?”
7. “Can it be adapted for children with special needs or for dual language learners?”
8. “Is it culturally appropriate?”
9. “Is it flexible enough to meet the needs of individual children and different learning styles?”
10. “Does it support children’s independence, ability to complete tasks and confidence?”

What are Coding Concepts?

A coding concept is a big idea in computer science that is widely applicable across situations, languages, and problems. The coding concepts you learned in this lesson are commands, loops (including nested loops), events, and conditionals.

What are Computational Thinking Concepts?

See Table 2. Components of Computational Thinking.

Why are Content Connections Important?

Regardless of your teaching domain, it is important to help your students make connections between coding and other academic content. This may be accomplished by explicitly pointing out the connection or, preferably, by using questioning techniques to elicit the learning from the students. Helping students make such connections can help them see the relevance of coding, transfer their learning across domains, and learn the other content more deeply (Rich, Jones, Belikov, Yoshikawa, and Perkins, 2017; Yadav, Stephenson, and Hong, 2017).

PIC-RAT Evaluation Model

The PIC-RAT evaluation model is a heuristic that facilitates the evaluation of any educational technology intervention. If you are not familiar with the PIC-RAT model, or if you need a refresher, please refer to the image below. (For additional information, please see Chapter 1: Effective Technology Integration in Royce Kimmons' open textbook K-12 Technology Integration [https://k12techintegration.pressbooks.com]).

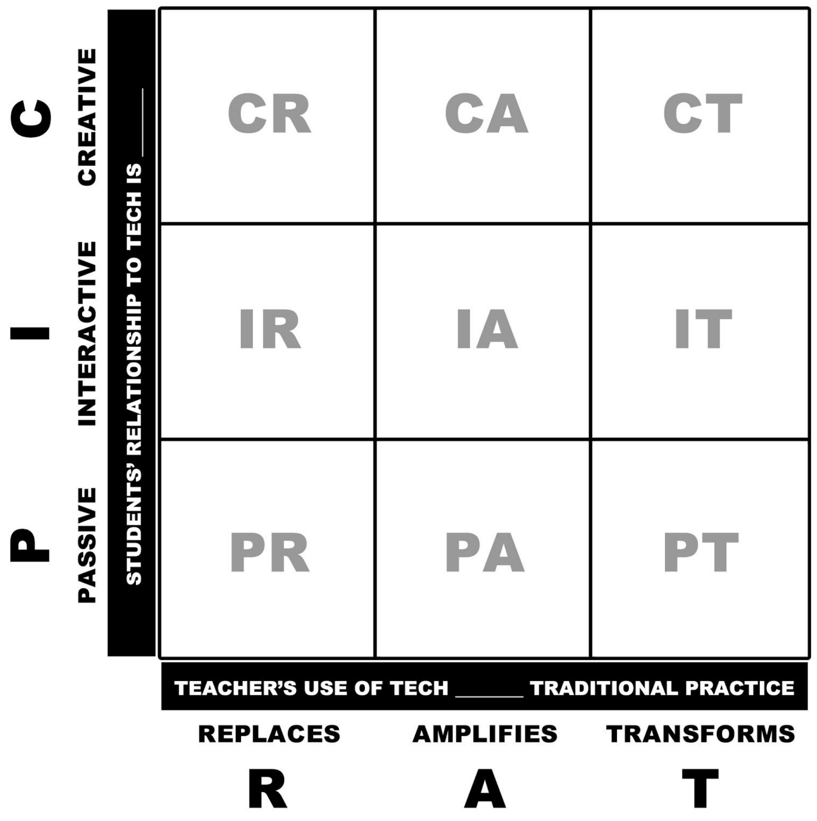

C
CREATIVE
I
INTERACTIVE
P
PASSIVE
STUDENTS' RELATIONSHIP TO TECH IS ____
CR
CA
CT
IR
IA
IT
PR
PA
PT
TEACHER'S USE OF TECH ____ TRADITIONAL PRACTICE
REPLACES
AMPLIFIES
TRANSFORMS
R
A
T

Image 2. PIC-RAT Matrix. Image by Royce Kimmons, CC-BY license.

Repository of Activities to Evaluate

You may use any resource you find online or (even better) any live lesson that you observe in which a codable robot is used in a learning activity. The following sample resources are provided for your convenience. Regardless of whether you use an activity from the list below, or one that you find on your own, do your best to make sure the activity falls within or close to your teaching domain.

Codable Robots - Sample Activities for Evaluation			
Resource	**Robot**	**Format**	**Domain**
Dash & Dot Teaching Kids to Code at Margo Elementary (https://youtu.be/OtIAnloKUNY)	Dash	Video	Upper Elementary

Kids Can Code: A Dot and Dash Addition Strategy Game (https://youtu.be/XluvddnVqmg)	Dash	Video	PreK-2
Robots in the Classroom (https://youtu.be/gloZqGnbtt4)	Dash	Video	PreK-2
Wonder League - robotics club for elementary (https://youtu.be/MjXBriJQlf8)	Dash	Video	3-6
Sprk Education Program - programming, robotics, and math for kids (https://youtu.be/EaNmAjxxIDU)	Sphero	Video	Middle School
Sphero SPRK Inspiring in the Classroom (https://youtu.be/0yQYr7ClxBc)	Sphero	Video	3-6
4th Grade Sphero Math (https://youtu.be/400ycM7nJxg)	Sphero	Video	3-6 Math
Hour of Code Sphero Challenge (https://youtu.be/eNL-4gjXT-Y)	Sphero	Video	Middle School
Selah Intermediate Ozobot Olympics (https://youtu.be/3Tz_S1A4VD0)	Ozobot	Video	3-6
First Grade Ozobots Coding 2016 (https://youtu.be/QEjSFCoxAvA)	Ozobot	Video	3-6
6th Grade Ozobot Trick or Treating (https://youtu.be/QeTsRFnSSog)	Ozobot	Video	6-8
Ozobot Adventure with Coding (https://youtu.be/OtUXrVGW3r0)	Ozobot	Video	6-8 Lang. Arts
What Educators have to Say About Ozobot (https://youtu.be/WQcwDgoEV-o)	Ozobot	Video	PreK-6
Bee Bots in the Classroom (https://youtu.be/wcAHpLO0BWA)	Bee Bot	Video	PreK-2
Bee-Bot®: a class introduction (https://youtu.be/52ZuenJlFyE)	Bee Bot	Video	PreK-2
Using Bee-Bot for Numeracy (https://youtu.be/za6wHl50fJU)	Bee Bot	Video	PreK-2 Numeracy

Using Bee-Bot for Literacy (https://youtu.be/ZJaSQgsDQ1w)	Bee Bot	Video	PreK-2 Literacy
BeeBot Skip Counting Activity (https://youtu.be/d3rFaICLFBw)	Bee Bot	Video	PreK-4
Karen McIntosh Bee Bots Learning in the Primary Classroom (https://youtu.be/QGg8GgWK0-s)	Bee Bot	Video	Elementary
Programming the Bee Bot (https://youtu.be/zFU3-nrSvsQ)	Bee Bot	Video	Elementary

Table 3. Codable Robots - Sample Activities for Evaluation

Evaluation Think-Aloud Demonstration

If you're having trouble with this assignment, you may want to watch this 3-minute video that demonstrates a thought process you might follow in this evaluation process.

Video 5. TWK Tutorial 1: Evaluating Robotic Learning Activities. Click, copy, or type this link into your browser to view: https://youtu.be/BmTtxydJwSQ. ***Note:*** *This video by IPT Instructor is released into the Creative Commons under a CC-BY license.*

Requirement 2.3: Codable Robot Learning Activity Plan

For this requirement, use **the same** codable robot that you used for requirement 2.2.

Create an activity to guide learning with the codable robot of your choice. The activity should meet the following criteria:

Learning Activity Criteria:

- *Is developmentally-appropriate for students within your teaching domain.*
- *Helps students learn or extend their learning of one or more of the coding concepts listed in requirement 1.1 (i.e., commands, loops, nested loops, events, conditionals).*
- *Helps students make connections between the coding and/or computational thinking principles they are learning and other academic content (e.g., science, math, engineering, language arts, social studies, etc.).*

On your submission form, include the following:

Submission Elements:

- Robot Used: *An indication of which robot this lesson plan is for. This should be the same robot for which you evaluated an activity in requirement 2.2.*
- Computational Learning Challenge: *A challenge that you would issue to students in order to guide their learning.*
- Scaffolding Artifacts: *A description of artifacts (if any) that you would give to students to scaffold their learning (e.g., worksheets, physical objects).*
- Student-Teacher Interaction Plan: *A description of how you would interact with the students in order to foster learning (e.g., what are you doing while they are working? What support do you provide when they get stuck?).*
- Solution Key: *The specific code (or an example of specific code) that would satisfy the computational learning challenge.*
- Criteria Justification: *A written justification for how the activity meets the criteria listed under the "Learning Activity Criteria" section above.*

Information and Resources for your Activity Plan

Robot Used:

You may use any codable robot for this activity. At the time of publication of this badge guide, the following robots are available for checkout from the BYU McKay School Tech

Lab for all BYU students and teachers at BYU's partner schools. Additional robots may be available when you are completing this badge. You can see the current list of available robots and learn how to reserve them by following this link (http://bit.ly/2HWCpv7). You can also visit the Robot Guides page (http://bit.ly/2L2bXBf) on the Tech with Kids website (http://bit.ly/2IO8wNL) to learn more about each of the following robots.

- Bee-Bot
- Ozobot
- Dash
- Sphero Sprk+

Computational Learning Challenge

The computational learning challenge is a coding task that the students need to do to get the robot to perform a certain action. Sometimes it may be accompanied by specific requirements, such as a limit to the number of blocks of code that students can use, or a requirement that they use a loop. Here are some examples:

- Make sphero draw a square with no more than 4 lines of code.
- Get Dash to do a dance. Make sure you use a loop.

Scaffolding Artifacts Description

A scaffolding artifact is anything that you will use to help the students learn the content they need to complete the learning challenge. For example, will they use a worksheet? Will there be physical objects like an inclined plane or obstacles that they need to navigate through? For this badge, you do not need to create the scaffolding artifacts, but you should describe what they would be and how you would use them in the activity.

Student-Teacher Interaction Plan

Describe how you will interact with the students. How, for example, will you give instructions for them to complete the learning activity? How will you provide the necessary scaffolding? How soon will you intervene when you see students struggling? How will you intervene? As you create your student-teacher interaction plan, keep in mind the following research-based best practices for teaching coding through robotics (Hunsaker, n.d.):

- **Modeling:** Teachers should set an example of learning by modeling their own understanding, learning, and progress in computational thinking. Especially in the early stages, they should also model the computational thinking process for students so they understand what the learning, reflection, and revision look like (Highfield, 2015).
- **Integrating.** Teachers should collaborate with other teachers to facilitate the completion of interdisciplinary culminating projects (Bers, Flannery, Kazakoff, and Sullivan, 2014).
- **Releasing Responsibility Gradually.** When teaching CT, educators should start with direct instruction, move to a simple guided activity, then issue an open-ended challenge or problem (Buss and Gamboa, 2017). Teachers should then continue to guide behavior, even while working/playing as a team (Highfield, 2015).
- **Encouraging.** Insofar as possible, teachers should provide "encouragement and problem-solving hints and tips," rather than outright answers (Buss and Gamboa, 2017).
- **Questioning.** Rather than providing answers directly, teachers should ask "probing questions" before, during, and after learning activities (Buss and Gamboa, 2017; See also Highfield, 2015) These questions should encourage students to reflect on their learning and might begin with phrases like the following (Buss and Gamboa, 2017):
    - "What if you were to…"
    - "How would you…"
    - "Have you considered…"
- **Fostering alternative problem-solving.** Teachers should promote alternative ways of modeling a problem (Buss and Gamboa, 2017), such as
    - Drawing out solutions on paper.
    - Discussing alternative solutions as teams.
    - Relating challenges to more familiar circumstances.
- **Using CT vocabulary across the curriculum** (Yadav, Mayfield, Zhou, Hambrusch, and Korb, 2014). This can reinforce students' understanding of the terms and

help them see their applicability across the curriculum and in daily life. For example, a teacher might refer to a set of rules or procedures as an “algorithm”; invite students to create an “abstraction” of how they feel; or emphasize that you are practicing “pattern recognition” skills.

Solution Key

The solution key is the specific code (or an example of specific code) that would satisfy the computational learning challenge. The code may be in whatever language makes the most sense (including a block-based language, or even just plain English that describes the blocks and their relationship). You might consider building the code on the mobile device that controls the robot, then taking a screenshot of it to fulfill this requirement.

Criteria Justification

See the information provided earlier under “Understanding the Evaluation Criteria.”

Requirement 2.4: Pedagogical Reflection

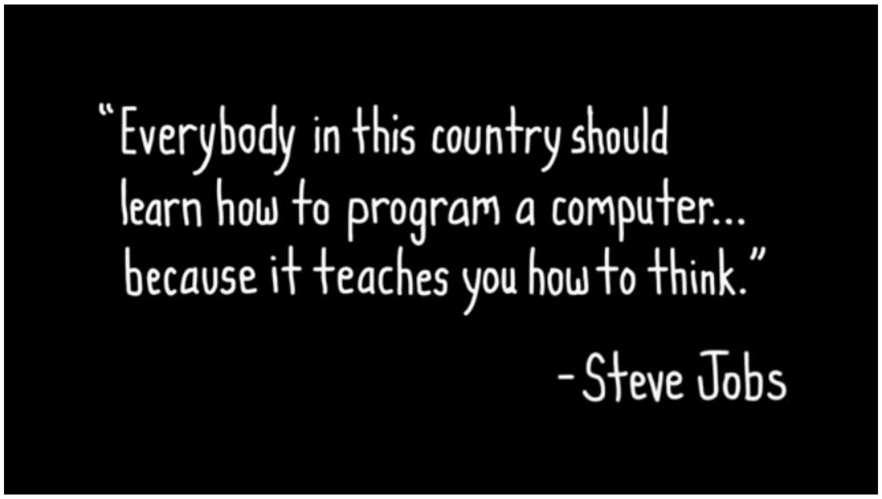
On your submission form, write a reflection in which you thoroughly respond to each of the following prompts (3-4 paragraphs):

1. *Provide a rationale for teaching the coding concepts highlighted in this badge (i.e., commands, loops, nested loops, events, conditionals) to students within your teaching domain.*
2. *What features of the codable robot you chose to complete this badge most appeal to you? Why?*
3. *In what ways could using this robot to teach coding in your classroom replace, amplify, and/or transform traditional classroom practices and settings? How could you use this technology to help students move from passive to interactive and creative uses of the technology? (See PIC-RAT framework).*
4. *Name at least three (3) specific computational thinking (CT) skills, attitudes, or approaches that you anticipate your codable robot learning activity plan (requirement 2.2) would foster. For each, justify thoroughly why you think that aspect of CT is supported by your plan.*

Before responding to the prompts in this reflection, it is highly recommended that you read or watch the following material that is applicable within your teaching domain:

Prompt 1: Rationale for Teaching Coding

To answer this prompt, watch the video and read one or two of the articles below. As you watch and read, consider how the arguments you encounter apply within your teaching domain. Also, think about why you were drawn to completing this badge. What about coding instruction do you think is relevant to your future as a k-12 educator?



“Everybody in this country should learn how to program a computer... because it teaches you how to think.”

-Steve Jobs

Video 6. What Most Schools Don’t Teach. Click, copy, or type this link into your browser to view: https://youtu.be/nKIu9yen5nc. ***Note:*** *This video is copyrighted by Code.org under a Standard YouTube License. It is therefore not included in the CC-BY license of the rest of this document.*

Articles to Peruse:

- 10 Reasons Kids Should Learn to Code
- 8 Reasons Why Kids Should Learn to Code
- 6 Reasons Why Your Child (and Mine!) Should Learn to Code
- Should All Children Learn to Code?

Prompt 2: Robot Features

To answer this prompt, consider paying special attention to why the robot of your choice is particularly fitted for use in your teaching domain.

Prompt 3: PIC-RAT

If you’re unfamiliar with or need a refresher on the PIC-RAT technology integration model, refer to the information provided earlier under the “PIC-RAT Evaluation Model” heading.

Prompt 4: Computational Thinking Connections

If you’re unfamiliar with or need a refresher on computational thinking, refer to the information provided earlier on Table 2 “Components of Computational Thinking.”

References

Bers, M.U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering : Exploration of an early childhood robotics curriculum. *Computers & Education, 72*, 145–157. https://doi.org/10.1016/j.compedu.2013.10.020

Buss, A., & Gamboa, R. (2017). Teacher transformations in developing computational thinking: Gaming and robotics use in after-school settings. In P.J. Rich & C.B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 189-203). Cham, Switzerland: Springer. Retrieved from http://sci-hub.cc/downloads/1d8d/10.1007@978-3-319-52691-1.pdf

[Code.org] (2013, February 6). *What most schools don't teach* [Video File]. Retrieved from https://youtu.be/nKIu9yen5nc

Hirschy, S. (2015). Developmentally appropriate technology in my classroom—But how? *Texas Child Care Quarterly, 39*(1). Retrieved from https://www.childcarequarterly.com/pdf/summer15_technology.pdf

Highfield, K. (2015). Stepping into STEM with young children: Simple robotics and programming as catalysts for early learning. In C. Donohue (Ed.), *Technology and digital media in the early years: Tools for teaching and learning* (pp. 150–161). New York, NY: Routledge.

Hunsaker, E. (n.d.). Integrating computational thinking. In R. Kimmons (Ed.) *K-12 technology integration.* Pressbooks. Retrieved from https://k12techintegration.pressbooks.com/chapter/integrating-computational-thinking/

Kimmons, R. (n.d.) Effective technology integration. In R. Kimmons (Ed.) *K-12 technology integration.* Pressbooks. Retrieved from https://k12techintegration.pressbooks.com/chapter/effective-technology-integration/

NAEYC, & Fred Rogers Center for Early Learning and Children's Media. (2012). *Technology and interactive media as tools in early childhood programs serving children from birth through age 8.* Retrieved from https://www.naeyc.org/sites/default/files/globally-shared/downloads/PDFs/resources/topics/PS_technology_WEB.pdf

Rich, P. J., Jones, B., Belikov, O., Yoshikawa, E., & Perkins, M. (2017). Computing and engineering in elementary school: The effect of year-long training on elementary teacher self-efficacy and beliefs about teaching computing and engineering. *International Journal of Computer Science Education in Schools, 1*(1), 1-20.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE)*, 14(1), 5.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 60*(4), 55–62. https://doi.org/10.1145/2994591