



GUIDE

PRACTICE GUIDE

Applying a Principled Approach to Develop and Use K-12 Computer Science Formative Assessments

Satabdi Basu
Daisy W. Rutstein
Carol Tate

Practice Guide: Applying a Principled Approach to Develop and Use K–12 Computer Science Formative Assessments

The National Comprehensive Center

The National Comprehensive Center (NCC) is one of 20 technical assistance centers supported under the U.S. Department of Education’s Comprehensive Centers program from 2019 to 2024. The NCC focuses on helping the 19 Regional Comprehensive Centers and state, regional, and local education agencies throughout the country to meet the daunting challenge of improving student performance with equitable resources.

This publication is in the public domain. While permission to reprint is not necessary, reproductions should be cited as:

Basu, S., Rutstein, D., & Tate, C. (2021). *Practice Guide: Applying a principled approach to develop and use K–12 computer science formative assessments*. National Comprehensive Center.

The contents of this publication were developed under a grant from the U.S. Department of Education. However, the contents do not necessarily represent the policy of the U.S. Department of Education, and you should not assume endorsement by the Federal government.

A copy of this publication can be downloaded from <https://www.nationalcompcenter.org/>.

Introduction

Formative assessment can be a powerful tool to support effective K-12 computer science (CS) instruction. Having teachers and students engage with the formative assessment process can increase student engagement and improve learning outcomes (Herman, 2013; Kingston & Nash, 2011; Popham, 2010; Wiliam, 2018). In this practice guide, we show how to apply the five-step process outlined in the corresponding [white paper](#) to systematically develop or select formative assessment tasks and use them to inform instruction. This guide illustrates formative assessment literacy in practice. We encourage school-level and state-level leaders to find ways to promote formative assessment literacy through CS professional development workshops, teacher communities of practice, policy guidelines, and other avenues. These five steps are discussed in greater detail:

- » **Step 1.** Define and/or identify fine-grained learning targets to assess.
- » **Step 2.** Determine evidence needed to measure progress towards learning targets.
- » **Step 3.** Find or create tasks that elicit the desired evidence.
- » **Step 4.** Determine how to evaluate and interpret the evidence provided by students.
- » **Step 5.** Relate the interpretation and/or evaluation of the evidence to possible follow-up activities.

This practice guide is not based on any specific curriculum; it can be used by anyone tasked with teaching CS. It is designed to enhance teachers' understanding of CS standards, determine how to select and implement appropriate formative assessment tasks, and learn how to modify instruction to address student challenges identified from the formative assessments. This process can increase teachers' knowledge of the CS content and how to teach it, as well as improve student engagement and learning.



Step 1: Define and/or Identify Fine-Grained Learning Targets to Assess

The first step in the process of developing and using CS formative assessments involves defining the fine-grained learning targets corresponding to specific CS lessons or activities and identifying which learning targets to focus on for formative assessment purposes. A learning target is a statement describing desired student proficiency in one specific aspect of a domain. This first step in the assessment design process comprises the following sub-steps:

- » Step 1a: Defining learning targets.
- » Step 1b: Identifying which learning targets to assess, and when to assess them.

Step 1a: Defining Fine-Grained Learning Targets

Teachers typically start with a CS curriculum or activity that has specified alignment with CS standards. Standards are generally expressed as a set of broad learning goals, each of which comprises multiple granular learning targets. Breaking down the standards highlights the critical aspects of student learning in the domain in a way that clarifies the expectations for instruction and assessment. In this practice guide, we choose a middle school CSTA standard (2-AP-12) and demonstrate how the standard can be decomposed into multiple fine-grained learning targets. The example we have chosen is a common learning objective in middle school.

2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals (Grades 6–8)

Control structures can be combined in many ways. Nested loops are loops placed within loops. Compound conditionals combine two or more conditions in a logical relationship (e.g., using AND, OR, and NOT), and nesting conditionals within one another allows the result of one conditional to lead to another. For example, when programming an interactive story, students could use a compound conditional within a loop to unlock a door only if a character has a key AND is touching the door.

This standard focuses on students' ability to use and develop nested control structures, such as nested loops, nested conditional statements, compound conditionals, nested procedures, conditionals nested within loops, and loops nested within conditionals.

Learning targets (fine-grained descriptions of what students should know and be able to do) for this middle school standard are shown below. Each learning target integrates a concept related to nested control structures with a computational practice. The concepts include nested loops, nested conditionals, compound conditionals, repeated conditionals, and procedures (a code module that performs a specific task and is referenced by a larger body of source code whenever required) inside control structures. Computational practices include Create (creating a computational artifact), Debug (debugging a computational artifact), and Interpret (interpreting the output of a



computational artifact). For example, learning target #1a integrates the concept of nested loops with the practices of creating and debugging computational artifacts.

Learning targets aligned with middle school standard 2-AP-12:

- » **1a. Nested loops + Create and debug:** Ability to create a nested loop and/or debug a given nested loop to represent a given scenario
- » **1b. Nested loops + Interpret:** Ability to recognize or identify the output of code that includes a nested loop. This includes the ability to identify how many times the action(s) will repeat and the sequence of repetition.
- » **2a. Nested conditionals + Create and debug:** Ability to create nested conditional statements and/or debug given nested conditionals to represent a given narrative description
- » **2b. Nested conditionals + Interpret:** Ability to identify the output of nested conditional statements
- » **3a. Compound conditionals + Create and debug:** Ability to create and/or debug a conditional statement that includes logical (AND, OR, NOT) operators to represent a given narrative description
- » **3b. Compound conditionals + Interpret:** Ability to identify the output of a conditional statement that uses logical (AND, OR, NOT) operators as part of the condition
- » **4a. Repeated conditionals + Create and debug:** Ability to create and/or debug a conditional statement that is evaluated repeatedly (conditional inside a loop, or a repeat-until construct)
- » **4b. Repeated conditionals + Interpret:** Ability to identify the output of a conditional statement that is evaluated repeatedly (conditional inside a loop, or a repeat-until construct)
- » **5a. Procedures inside a control structure + Create and debug:** Ability to create and/or debug a procedure that is called within a different control structure (e.g., procedure inside a loop, procedure called by another procedure)
- » **5b. Procedures inside a control structure + Interpret:** Ability to identify the output of a procedure that is called within a control structure (e.g., procedure inside a loop, procedure called by another procedure)

The middle school standards, including 2-AP-12, assume that students are proficient on the elementary level CS standards. With many students being introduced to CS for the first time in middle school, this is often not the case. Teachers working toward 2-AP-12 will need to assess students for prerequisite skills and focus on building these skills first where they are lacking.

Prerequisite elementary-level learning targets for middle school standard 2-AP-12:

- » **1a. Loops + Create and debug:** Ability to create and/or debug a loop that sets up an action that repeats, either forever or for a set number of times



- » **1b. Loops + Interpret:** Ability to recognize or identify the output of code that includes a simple loop. This includes the ability to identify how many times action(s) will repeat and what sequence they will repeat in.
- » **2a. Simple conditionals + Create and debug:** Ability to create and/or debug a simple conditional statement (IF-THEN or IF-THEN-ELSE)
- » **2b. Simple conditionals + Interpret:** Ability to identify the output of a simple conditional statement (IF-THEN or IF-THEN-ELSE)

Teachers can choose to begin with assessing the prerequisite elementary-level learning targets or move directly to assessing the middle school learning targets. Step 1b provides additional guidelines for choosing learning targets to assess.

Step 1b. Identify Which Learning Targets to Assess and When

Once teachers have determined the learning targets corresponding to the standards covered in their target curriculum materials, the next step is to identify which of them to assess and at what point during the lesson to assess them. Some learning targets, though important components of a standard, may not be the focus of assessments in specific classrooms or grade levels. When a curriculum activity claims to be aligned to a particular standard, it does not necessarily mean that the activity is aligned to all learning targets included as part of the standard. Teachers can take their daily lesson plans from the CS curriculum or activity they are using in their classrooms and map the lessons to specific learning targets.

The mapping between daily lessons and learning targets can help teachers visualize the flow of their instruction and identify when they want to insert formative assessment activities. There is no one way to determine when it is best to use formative assessments, but there are some key questions to consider:

1. Should learning targets be assessed separately at different time points, or should multiple learning targets be grouped together in a single assessment activity? Both approaches can be appropriate based on the time available and the focus of a certain course or curriculum—the important thing is to be aware of whether the activity elicits evidence about all the targets of interest.
2. Can formative assessment activities be integrated into existing classroom routines, such as daily exit tickets or classroom polls, instead of carving out separate time for assessment activities? Making assessment a regular part of instruction enables closer monitoring of progress and timely adjustments.
3. Are there learning targets that are critical for students to master in order to learn subsequent concepts? If yes, plan formative assessment activities after instruction on such learning targets. This way, a teacher can identify students who have not mastered a concept and may struggle with learning the next concept and find ways to support such students.

In steps 2–5, we focus on **Learning Targets 3a and 3b** related to compound conditionals to demonstrate how to design and use formative assessments aligned to learning targets.



Step 2: Determine Evidence Needed to Measure Progress Towards Learning Targets

Once a learning target has been identified, the next step is to determine what evidence can be gathered that would reflect on the learning target. The evidence must be something observable, such as a written or oral response. Thinking through what this evidence might look like helps ensure alignment between the assessment task and the learning targets. Below are examples of the types of evidence that would be appropriate for two of the learning targets listed in Step 1.

Learning Target 3a: Ability to create and/or debug a conditional statement that includes logical (AND, OR, NOT) operators to represent a given narrative description

This learning target can be further broken down into two parts: being able to *create* a conditional statement and being able to *debug* a conditional statement. When identifying the possible evidence, both aspects of the learning target should be considered.

Evidence that students know how to create a conditional statement for a scenario might be that they generate a conditional statement that uses AND, OR, or NOT and that matches the description of the scenario, or that they identify a correct conditional statement from a list of options.

Expectations concerning the degree and depth to which students are engaging with the learning target may vary and may change over time.

For the debugging aspect of the target, teachers must decide if it is enough for students to identify an error or if students should also be required to explain why the error occurred and/or fix the error. Again, there is a range of possible evidence that a student might provide, and it is up to the teacher to decide what is most appropriate based on what they want to know about their students. Teachers may also want to consider if they are measuring debugging along with creating, and if so, what additional evidence they would need to make sure the student is engaging in both practices.

Examples of possible evidence for Learning Target 3a:

- » Students create a computer program that includes a conditional statement with a logical operator to match a given scenario.
- » Students select an appropriate conditional statement with a logical operator from given options to match a given scenario.
- » Students specify the logical operator and variables within a conditional to match a given scenario.
- » Students state whether or not a conditional statement matches a given scenario.
- » Students explain why a given conditional statement does not match a given scenario.
- » Students select from given options to identify what would have to change to make a conditional statement match a given scenario.



- » Students modify a given conditional statement to match a given scenario.
- » Students describe how to modify a conditional statement to match a given scenario

Learning Target 3b: Ability to identify the output of a conditional statement that uses logical (AND, OR, NOT) operators as part of the condition

This learning target is about students demonstrating that they understand the logic in a conditional statement involving logical operators. Teachers may provide possible outputs (e.g. as a multiple-choice item) or they may require students to predict the output of conditional statements. Teachers should decide which of these types of evidence aligns with their goals and choose or design a task that will elicit the kind of evidence they seek.

Possible evidence for Learning Target 3b:

- » Students select the output of a conditional statement that uses logical operators from a given set of options.
- » Students generate the output of a conditional statement that uses logical operators.
- » Students state whether or not a given conditional statement involving logical operators produces a given output.



Step 3: Find or Create Tasks that Elicit the Desired Evidence

In this step, teachers develop or select appropriate assessment tasks corresponding to learning targets. Specifying the task characteristics, or requirements of all assessment tasks, first will help ensure that the tasks produce the desired evidence of learning. The task characteristics are related to the learning target but may also include other task requirements (e.g., reading level or amount of reading). Below are some examples of task characteristics.

Examples of task characteristics for Learning Target 3a:

- » Task includes a scenario that would need a conditional statement with at least one logical operator.
- » Task includes a specific representation or programming environment in which to create or debug a conditional statement.
- » For debugging, task includes a conditional statement that has an error in it.
- » Task includes a prompt asking student to create a conditional statement or identify and correct the error in a conditional statement.
- » Task is written using simple language for a reading level that is a grade below to ensure that students are able to comprehend the language used in the task.

Examples of task characteristics for Learning Target 3b:

- » Task includes a conditional statement (in a given representation) with at least one logical operator.
- » Task includes a prompt asking students to identify the results of the conditional statement given a set of conditions.

Once task characteristics are determined, tasks can be selected or developed to match the characteristics. Pre-existing tasks can be selected if they align with the identified task characteristics. When developing assessment tasks, the context in which the tasks will be used and the desired level of complexity need to be considered. The scenario described in a task needs to be relatable and meaningful for its intended audience. The level of complexity can be varied based on the type of evidence sought by a teacher, as described in Step 2. For example, for Learning Target 3a, students can be asked to create a conditional statement on their own, fill in specific parts of an incomplete conditional statement, or select from given options for how to create a conditional statement. Varying the number of logical operators required in a conditional statement is another way to vary the complexity of assessment tasks. Assessing more than one learning target simultaneously as part of an assessment task is yet another way of increasing task complexity. Even for a single learning target, it is important to ensure that factors external to the learning target like reading load or knowledge of math operators like “<” and “>” do not interfere with the assessment of the learning target.

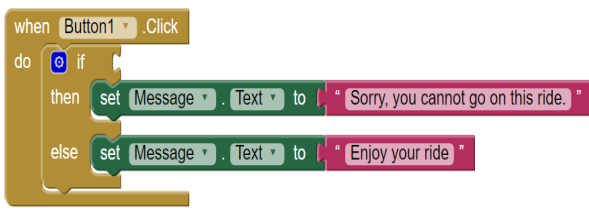


Sometimes, constraints related to scoring assessment also need to be considered when developing or selecting tasks. For example, automatic scoring may be required, or the time needed to score may need to be minimized. Open-ended responses may require the development of a scoring rubric, some discussion and agreement among teachers about the elements of a correct and complete response, and hence more time to score. Multiple-choice items, on the other hand, take much less time to score than open-ended responses and scoring can be easily automated, though they require more work to create up front. The decision of which item format to use also depends on existing classroom norms and the familiarity of students and teachers with different formats. For example, a teacher may use a multiple-choice task along with a method for having students assign a confidence rating to the answer choices. This format provides teachers with additional information on how confident students are in their responses as well as what other answer options students were considering. However, students need to be comfortable accurately portraying their confidence rating and need to have an understanding of how this rating will be used. Teachers using the confidence rating along with multiple-choice questions also need to understand how to combine the confidence information along with student responses to elicit evidence about student learning.

We illustrate some examples of assessment tasks for Learning Targets 3a and 3b in Figures 1– 4.



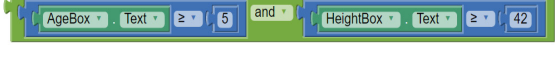

Figure 1. A multiple-choice assessment task aligned to Learning Target 3a¹

A water coaster ride at an amusement park has age and height requirements. You have to be at least 5 years old and at least 42 inches tall to be able to go on the ride. The following app is written to check if someone is able to take the ride.



```
when Button1 .Click
do
  if
  then
    set Message .Text to "Sorry, you cannot go on this ride."
  else
    set Message .Text to "Enjoy your ride."
```

Which of the following expressions should be attached to the *if* block?

- ☐ 
- ☐ 
- ☐ 
- ☐ 

¹ Assessment developed for CoolThink@JC initiative. Suggested citations: (1) Basu, S., Rutstein, D. W., Xu, Y., Wang, H., & Shear, L. (2021). A principled approach to designing computational thinking concepts and practices assessments for upper elementary grades. *Computer Science Education*, 1-30; (2) Shear, L., Wang, H., Tate, C., Basu, S., & Laguarda, K. (May, 2020). *CoolThink@JC pilot evaluation: Endline report*. SRI International.

Figure 2. The basketball task - a constructed response assessment task aligned to Learning Target 3a

Adrina is creating a computer game in which a player gets three chances to get a ball into a basket. After the player makes 3 attempts, the game tells the player if they won or not.


There are two ways to win:

- Getting all 3 balls into the basket
- Getting at least 1 ball into the basket from a distance of more than 20 feet away.

Adrina uses the following variables in her code:

- NumBaskets – the number of balls the player gets into the basket
- Distance - the greatest distance of a ball that makes it into the basket |


She programs the following code to decide who wins:



Sam makes 2 baskets but misses the third basket. One of the baskets was made from a distance of 23 feet, and the other from a distance of 28 feet. Adrina's program should tell Sam "You win!" but does not. How can you fix Adrina's program?

Figure 3. A multiple-choice assessment task aligned to Learning Target 3b¹

A teacher uses the following code to print out stickers for her students based on students' quiz scores.



Tina scored 45, Yi scored 50, Liu scored 70 and Lily scored 80. What stickers would each of them receive?

☐ Tina and Yi: "Good Job" stickers, Liu and Lily: "Exceptional" stickers

☐ Yi and Liu: "Good Job" stickers, Tina and Lily: "Exceptional" stickers

☐ Tina, Yi and Liu: "Good Job" stickers, Lily: "Exceptional" sticker

☐ Liu: "Good Job" sticker, Tina, Yi and Lily: "Exceptional" stickers

¹ Assessment developed for CoolThink@JC initiative. Suggested citations: (1) Basu, S., Rutstein, D., Xu, Y., & Shear, L. (2020). A Principled Approach to Designing a Computational Thinking Practices Assessment for Early Grades. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (pp. 912-918); (2) Shear, L., Wang, H., Tate, C., Basu, S., Laguarda, K. (May, 2020). CoolThink@JC Pilot Evaluation: Endline Report. Menlo Park, CA: SRI International..

Figure 4. A multiple-choice assessment task aligned to Learning Target 3b that does not use any specific programming language representation

Consider the following pseudocode snippet:

If the apple is green and the peanut is shelled,

- *Move the apple up 2 steps*
- *Move the peanut down 2 steps*

Otherwise, move the peanut up 2 steps.

Under which of the following sets of sizes/positions would the peanut move up two steps? Select all that apply.

- A. The apple is green, and the peanut is shelled
- B. The apple is green, and the peanut is unshelled
- C. The apple is yellow, and the peanut is shelled
- D. The apple is yellow, and the peanut is unshelled



Step 4: Determine How to Evaluate and Interpret the Evidence Provided by Students

When selecting or developing a task, it is important to consider how the task will be evaluated and what inferences will be made about a student based on that evaluation. In this next step, teachers will generate a list of the type of evaluation criteria desired. Teachers should determine how they want to evaluate the evidence that was specified in Step 2. Once the task is developed/selected, teachers will use the evaluation criteria to create a rubric that applies a set of scoring rules to the task. Tables 1 and 2 show examples (not an exhaustive list) of possible evaluation criteria and corresponding evidence statements for Learning Targets 3a and 3b respectively.

Table 1. Examples of possible evaluation criteria for Learning Target 3a - creation and debugging of compound conditionals

Evidence Statement (from Step 2)	Examples of evaluation criteria
Students create a computer program that includes a conditional statement with a logical operator to match a given scenario	How well the computer program matches the given scenario
Students select an appropriate conditional statement with a logical operator from given options to match a given scenario	Whether the conditional statement selected from the given options is correct. OR Whether the conditional statement selected matches a common student challenge
Students state whether or not a conditional statement matches a given scenario	Ability to identify cases when the conditional correctly models a given scenario and cases when the conditional fails
Students explain why a given conditional statement does not match a given scenario Students modify a given conditional statement to match a given scenario	Accuracy of the explanation of why a conditional statement does not match a given scenario Accuracy of the correction(s) to a given compound conditional statement.

Table 2. Examples of possible evaluation criteria for Learning Target 3b – interpretation of compound conditionals

Evidence Statement (from Step 2)	Examples of evaluation criteria
Students select the output of a conditional statement that uses logical operators from a given set of options	Whether or not the correct output was selected
Students generate the output of a conditional statement that uses logical operators	Accuracy of the generated or predicted output
Students state whether or not a given conditional statement involving logical operators produces a given output	Whether or not the student correctly identified if a given conditional statement produces the given output



Consideration of the possible types of evidence desired can help point to the type of task that should be used. For example, for the basketball task described in Figure 2 aligned to Learning Target 3a, students are given a scenario and are shown corresponding code. The task focuses on assessing students' ability to debug a conditional statement, and provides the information that an error occurred, thus focusing the task on having students modify the provided code. The target evaluation criterion in this scenario is "Accuracy of the correction(s) to a given compound conditional statement." Determining how to score students' open-ended responses depends on what constitutes an accurate fix for the given error. In this case, what a teacher might look for includes the following:

- » The student indicates that the current program can be fixed by adding or modifying code to work with 2 baskets (when numBaskets = 2)
- » The student describes how the current program can be modified so that the program says "You win" to Sam who has made 2 baskets.
- » The student does not introduce new errors that would make the program provide incorrect responses for scenarios where it now provides the correct response.

A teacher can create a rubric that looks for these three aspects. The rubric can specify how to apply the checklist of aspects to the student work and get a score, or it can help categorize students depending on the types of errors they make. How a teacher designs the rubric depends on how they are using the information from the task and their follow-up strategies. Table 3 shows an example rubric with example student responses.



Table 3. A sample rubric corresponding to assessment task in Figure 2

Score	Rubric criteria	Student Response
3 points	<ul style="list-style-type: none"> » Student indicates the code needs to be fixed to work with 2 baskets. » Student modifies the code to work when the player makes 2 baskets » Student’s code continues to work in all other cases. 	<ul style="list-style-type: none"> » The code needs to change to make it so you can win with 2 baskets. You can change the <code>=1</code> to be <code>>= 1</code> and that would work. » The code needs to add the case for 2 baskets. It needs to add another conditional statement that says if the baskets = 2 and the distance is greater than 20 then also return you win.
2 points	<ul style="list-style-type: none"> » Student indicates that the code needs to be fixed to work with 2 baskets. » Student modifies the code to work when the player makes 2 baskets, but introduces a new error that causes at least 1 other case not to work. 	<ul style="list-style-type: none"> » Can change the code to make it okay for 2 baskets. Just change it so all it has is <code>(NumBaskets>0 and distance >20)</code>. [Student misses the case where the distance is not greater than 20 and the person makes 3 baskets].
1 point	<ul style="list-style-type: none"> » Student indicates that the code needs to be fixed to work with 2 baskets. » Student does not describe a fix or modifies the code in a way that does not fix the error. 	<ul style="list-style-type: none"> » You have to fix the program for the case where NumBaskets is 2.
0 point	<ul style="list-style-type: none"> » Student does not respond, or the response does not indicate how to fix the error in the code. 	<ul style="list-style-type: none"> » The code needs to be fixed.

If the teacher was interested in additional information, such as whether or not the student could identify the error, they may modify the assessment task prompt and add additional questions. For example, the teacher could remove the part of the question that specifies that the program has an error and instead ask the student to identify whether the program functions correctly or has an error, as shown in Figure 5.

For rubric development for this new version, the first step is again going back to the evidence statement generated in Step 2. With the addition of questions **a** and **b**, the task is also addressing the evidence statement “Students generate the output of a conditional statement that uses logical operators.” Related to this evidence statement is the evaluation statement: Accuracy of the generated or predicted output. While the correct response for question **a** of the basketball task is “You did not win”, a teacher may not only care about correct versus incorrect responses, but also about what incorrect response a student chooses. If a teacher only wants to know about the correctness of student responses, they can just check whether or not the student selected or generated the correct response. If a teacher wants to know more about students’ thinking and why students got the response incorrect, they can categorize students by their responses to question **a**. Students who choose an option where the program says “You win” might have difficulties navigating compound conditionals, or distinguishing between what a program should do and what

it actually does, while students who picked “It does not say anything” or “It gives an error message” might have general confusion about how conditional statements work.

Figure 5. Alternate version of the basketball task from Figure 2

Adrina is creating a computer game in which a player gets three chances to get a ball into a basket. After the player makes 3 attempts, the game tells the player if they won or not.


There are two ways to win:

- Getting all 3 balls into the basket
- Getting at least 1 ball into the basket from a distance of more than 20 feet away.

Adrina uses the following variables in her code:

- NumBaskets – the number of balls the player gets into the basket
- Distance – the greatest distance of a ball that makes it into the basket

She programs the following code to decide who wins:



```
if (NumBaskets = 3 or NumBaskets = 1 and Distance > 20) then
  say "You win!"
else
  say "You did not win!"
```

a. Sam makes 2 baskets but misses the third basket. One of the baskets has a distance of 23 and the other has a distance of 28. What would the program say after Sam made the above 3 attempts?

- ☐ “You win!”
- ☐ “You did not win.”
- ☐ It does not say anything
- ☐ It gives an error message

b. Does what the program say match what it is supposed to say?

- ☐ Yes, what the program says is correct
- ☐ No, there is an error in the program

c. If you answered Yes to part b, explain why this is correct. If you answered No to part b, how would you fix the program?

Step 5: Relate the Interpretation and/or Evaluation of the Evidence to Possible Follow-up Activities

In the final step, teachers determine how to interpret and respond to the evidence they have gathered. The goal of formative assessment is to make instruction responsive to student thinking. This means taking a careful look at the mistakes that students make. The specific nature of student mistakes drives the feedback. Sometimes, mistakes are due to random guesses or one-off slips, but more often they reveal underlying misconceptions (confidence indicators associated with assessments can help distinguish between errors due to random guesses and errors due to misconceptions). It is important to understand that mistakes are often related to students' prior knowledge. Understanding the origin of the mistake – whether it stems from a flawed idea of how computer programs work or from an overapplication of an idea from another discipline like mathematics — can be helpful for teachers.

After identifying a student misconception, a teacher's next steps can include re-teaching the topic, providing individual feedback, or facilitating whole-class discussions, code-tracing activities, argumentation activities, and additional unplugged or programming activities focusing on similar concepts. Several factors will influence how a teacher proceeds, including how much time can be allotted, how widespread the learning challenge seems to be, and what additional activities are available. Sometimes, re-teaching a concept to the whole class is warranted; sometimes the group is ready to move on, but a teacher may need to follow up with a few students. Establishing classroom routines that enable the efficient use of formative assessment data is important. For example, making student thinking visible to the class through projecting student responses and answer choices or facilitating collective code-tracing activities on the whiteboard can inspire productive class discussions. Pairing students who give opposing responses can also be a good way to provoke deeper thinking. Table 4 provides examples of how the evidence from the alternate version of the basketball task (Figure 5) can be interpreted and provides possible next steps.



Table 4. Example follow-up steps for student responses to the alternate version of the basketball task shown in Figure 5

Interpretation of Student Responses	Follow-up Steps
Most students answer questions a , b , and c correctly	Move on to the next topic or lesson and follow up with a few students individually
Students who answer question a with “you win!” and question b with “Yes, what the program says is correct” are showing that they understand how the program should work, but cannot comprehend how the program actually does work. These students are not able to identify the error in the program. This could be a case of the “superbug,” (Pea et al., 1987), the expectation that the computer will use common sense (i.e., if one basket wins, then two should also win). These students will benefit from follow-up activities that make clear how computers and humans “think” differently.	Have students write directions for each other to follow exactly (such as how to make a peanut butter and jelly sandwich) to help students understand that computers follow directions, whether they make sense humanly or not.
Students who answer question a with “you win!” and question b with “No, there is an error in the program” are showing that they neither understand how the program is supposed to work nor can they comprehend how the program actually works.	First, have students come up with different ways to win the basketball game. Help students understand the scenario in question a and that Sam should win the game. Then have students trace the code or embody different variables to predict the output of the code segment.
Students who answer question a correctly with “You did not win” are showing that they understand how the conditional statement works. If these students are asked to share their response to question c about fixing the program, it can initiate rich discussions about different approaches to solving the problem. Students can reason about why some approaches work and some do not, and how some approaches are more elegant and efficient than others. Looking for patterns in student approaches to debugging can also yield insight into student misconceptions and thus guide follow-up instruction.	Once a student has suggested a fix, have other students trace the resultant code block-by-block (or line-by-line) to determine whether the suggested fix works or not.
Students who expect the computer to not say anything or give them an error message in this scenario are showing that they do not understand how conditional statements work. In this case, revisiting the fundamental properties of conditionals, perhaps with unplugged activities at first, would be a good next step.	Play a game with students where you read a series of “if/then/else” statements and they stand or sit depending on whether the statement is true or false. Start with simple conditions and be sure to later include multiple conditions joined through logical operators.



Conclusion

The goal of the 5-step process outlined above is to ensure that the formative assessment matches its purpose and meets the teacher’s need. The information specified in each of the steps clarifies the requirements of the assessment and specifies how the assessment can be used. By using this process, teachers can both support their students’ learning and deepen their own understanding of the CS standards and how students are expected to engage with these standards.

References

- Herman, J. (2013). *Formative assessment for next generation science standards: A proposed model*. Invitational Research Symposium on Science Assessment. CRESST.
- Kingston, N. & Nash, B. (2011). Formative assessment: A meta-analysis and a call for research. *Educational Measurement: Issues and Practice*, 30, 28–37.
- Pea, R.D., Soloway, E., & Spohrer, J. C. (1987). The buggy path to the development of programming expertise. *Focus on Learning Problems in Mathematics*, 9(1), 5–30. <https://hal.archives-ouvertes.fr/hal-00190540/en/>
- Popham, W. J. (2009). Assessment literacy for teachers: Faddish or fundamental? *Theory Into Practice*, 48(1), 4–11.
- William, D. (2018). *Embedded formative assessment*. Solution Tree Press.

