Tips for Assessment

CSTeachingTips.org/Tips-for-Assessing-Programming

1. **Predict the output of code**
   to demonstrate code tracing ability.
   Draw a diagram as you trace through the lines of code!

2. **Find and fix a bug in code**
   to demonstrate debugging skills.
   Programming involves a lot of debugging! Let's practice!

3. **Explain, compare, or critique code**
   to practice abstracting from lines of code.
   Describe the code to someone who has never seen code.

4. **Arrange code segments**
   to code without syntax errors.
   Focus on your code's logic by rearranging these lines!

5. **Solve the problem by hand**
   to demonstrate understanding of an algorithm.
   Before coding, test your understanding of the algorithm.

6. **Create a portfolio**
   to show off the breadth of their skills.
   Show your friends and family all that you've learned!

7. **Write or modify code**
   to demonstrate programming fluency.
   Let's practice all of the skills at the same time!

csteachingtips

1 Predict the output of code

Ask students what code will output when provided specific inputs. You can also reverse this and ask students to provide inputs that will produce specific outputs. Ask students to compare the output of code in different programs to help students see the differences between similar concepts or commands. Help students understand that being able to predict the behavior of commands in a programming language is an important prerequisite to being able to write code. For all of these you can use them as formative assessment in class.

2 Find and fix a bug in code

Ask students to identify a bug in code. You can model good debugging practices by showing various inputs that produce correct and incorrect output. You can ask students to fix a bug that you demonstrate with tests or ask them to find a bug. For example, you can ask students to write a test case that will demonstrate the bug.

3 Explain, compare, or critique code

Ask students to write sentences to describe code in a way that a friend that isn't in the class would understand. This provides students the opportunity to demonstrate that they can abstract from individual lines of code. You can make the task a little easier by telling students to summarize the responsibility or behavior of particular parts of the code. As an easier to grade alternative, you can ask students to rename variables in the code.

4 Arrange code segments

Give students a set of lines of code and ask students to order them to produce a program with specific behavior. These problems are typically called "Parson's Problems" and allow students to reason about the logic of the code without having to worry about syntax. You can make this more difficult by including extraneous lines of code that students don't need to solve the problem.

5 Solve the problem by hand

Before students try to write code that solves a problem, make sure that they can solve the problem by hand. This can involve writing test cases that show that they can predict the expected behavior. You can also have students describe or draw the output of an algorithm.

6 Create a portfolio

It is difficult to write assessments that capture the breadth of the skills students have learned. Have students create a portfolio that shows off their work throughout the class. This can also be motivating for students to be able to see and share what they learned in the class! You can also give students a rubric for the skills they need to demonstrate within a project or their portfolio.

7 Write or modify code

It probably goes without saying that to assess students' ability to write code, you could ask them to write code. Additionally, consider providing students code that they need to modify to change the behavior of the code. This can provide students practice reading code and identifying the important features that determine each behavior.