UNIT 1 ACTIVITY

DEBUG IT!

SUGGESTED TIME
15-30 MINUTES

OBJECTIVES

By completing this activity, students will:

+ investigate the problem and find a solution to five debugging challenges
+ explore a range of concepts (including sequence) through the practices of testing and debugging
+ develop a list of strategies for debugging projects

ACTIVITY DESCRIPTION

- ❑ Optionally, have the Unit 1 Debug It! handout available to guide students during the activity.
- ❑ Help students open the Debug It! programs from the Unit 1 Debug It! studio or by following the project links listed on the Unit 1 Debug It! handout. Encourage students to click on the "Look Inside" button to investigate the buggy program, tinker with problematic code, and test possible solutions.
- ❑ Give students time to test and debug each Debug It! challenge. Optionally, have students use the remix function in Scratch to fix the bugs and save corrected programs.
- ❑ Ask students to reflect back on their testing and debugging experiences by responding to the reflection prompts in their design journals or in a group discussion.
- ❑ Create a class list of debugging strategies by collecting students' problem finding and problem solving approaches.

RESOURCES

- ❑ Unit 1 Debug It! handout
- ❑ Unit 1 Debug It! studio
  http://scratch.mit.edu/studios/475483

REFLECTION PROMPTS

+ What was the problem?
+ How did you identify the problem?
+ How did you fix the problem?
+ Did others have alternative approaches to fixing the problem?

REVIEWING STUDENT WORK

+ Were students able to solve all five bugs? If not, how might you clarify the concepts expressed in the unsolved programs?
+ What different testing and debugging strategies did students employ?

NOTES

+ This activity works well in groups! Get students working in teams of 2-4 people to collectively problem solve and share debugging strategies.
+ Testing and debugging is probably the most common activity of programmers. Things rarely work as planned, so developing a set of testing and debugging strategies will be beneficial to any computational creator.

NOTES TO SELF

- ❑ .....
- ❑ .....
- ❑ .....
- ❑ .....

UNIT 2 ACTIVITY

DEBUG IT!

SUGGESTED TIME
15-30 MINUTES

OBJECTIVES

By completing this activity, students will:

+ investigate the problem and find a solution to five debugging challenges
+ explore a range of concepts (including sequence and loops) through the practices of testing and debugging
+ develop a list of strategies for debugging projects

ACTIVITY DESCRIPTION

- ❑ Optionally, have the Unit 2 Debug It! handout available to guide students during the activity.
- ❑ Help students open the Debug It! programs from the Unit 2 Debug It! studio or by following the project links listed on the Unit 2 Debug It! handout. Encourage students to click on the "Look Inside" button to investigate the buggy program, tinker with problematic code, and test possible solutions.
- ❑ Give students time to test and debug each Debug It! challenge. Optionally, have students use the remix function in Scratch to fix the bugs and save corrected programs.
- ❑ Ask students to reflect back on their testing and debugging experiences by responding to the reflection prompts in their design journals or in a group discussion.
- ❑ Create a class list of debugging strategies by collecting students' problem finding and problem solving approaches.

RESOURCES

- ❑ Unit 2 Debug It! handout
- ❑ Unit 2 Debug It! studio
  http://scratch.mit.edu/studios/475539

REFLECTION PROMPTS

+ What was the problem?
+ How did you identify the problem?
+ How did you fix the problem?
+ Did others have alternative approaches to fixing the problem?

REVIEWING STUDENT WORK

+ Were students able to solve all five bugs? If not, how might you clarify the concepts expressed in the unsolved programs?
+ What different testing and debugging strategies did students employ?

NOTES

+ Facilitate this activity in a whole group by having students act out the Debug It! programs in a similar way to the Performing Scripts activity, or introduce performing scripts as a new strategy for testing and debugging projects.

NOTES TO SELF

- ❑ .....
- ❑ .....
- ❑ .....
- ❑ .....

UNIT 3 ACTIVITY

DEBUG IT!

SUGGESTED TIME
15–30 MINUTES

OBJECTIVES

By completing this activity, students will:

+ investigate the problem and find a solution to five debugging challenges
+ explore a range of concepts (including events and parallelism) through the practices of testing and debugging

ACTIVITY DESCRIPTION

- ❑ Optionally, have the Unit 3 Debug It! handout available to guide students during the activity.
- ❑ Help students open the Debug It! programs from the Unit 3 Debug It! studio or by following the project links listed on the Unit 3 Debug It! handout. Encourage students to click on the "Look Inside" button to investigate the buggy program, tinker with problematic code, and test possible solutions.
- ❑ Give students time to test and debug each Debug It! challenge. Optionally, have students use the remix function in Scratch to fix the bugs and save corrected programs.
- ❑ Ask students to reflect back on their testing and debugging experiences by responding to the reflection prompts in their design journals or in a group discussion.
- ❑ Create a class list of debugging strategies by collecting students' problem finding and problem solving approaches.

RESOURCES

- ❑ Unit 3 Debug It! handout
- ❑ Unit 3 Debug It! studio
  http://scratch.mit.edu/studios/475554

REFLECTION PROMPTS

+ What was the problem?
+ How did you identify the problem?
+ How did you fix the problem?
+ Did others have alternative approaches to fixing the problem?

REVIEWING STUDENT WORK

+ Were students able to solve all five bugs? If not, how might you clarify the concepts expressed in the unsolved programs?
+ What different testing and debugging strategies did students employ?

NOTES

+ Being able to read others' code is a valuable skill and is critical for being able to engage in the practices of reusing and remixing.
+ This activity is a great opportunity for pair programming. Divide students into pairs to work on the debugging challenges.
+ Students can explain their code revisions by right-clicking on Scratch blocks to insert code comments.

NOTES TO SELF

- ❑ .....
- ❑ .....
- ❑ .....
- ❑ .....

UNIT 4 ACTIVITY

DEBUG IT!

SUGGESTED TIME
15-30 MINUTES

OBJECTIVES

By completing this activity, students will:

+ investigate the problem and find a solution to five debugging challenges
+ explore a range of concepts (conditionals, operators, and data) through the practices of testing and debugging

ACTIVITY DESCRIPTION

- ❑ Optionally, have the Unit 4 Debug It! handout available to guide students during the activity.
- ❑ Help students open the Debug It! programs from the Unit 4 Debug It! studio or by following the project links listed on the Unit 4 Debug It! handout. Encourage students to click on the "Look Inside" button to investigate the buggy program, tinker with problematic code, and test possible solutions.
- ❑ Give students time to test and debug each Debug It! challenge. Optionally, have students use the remix function in Scratch to fix the bugs and save corrected programs.
- ❑ Ask students to reflect back on their testing and debugging experiences by responding to the reflection prompts in their design journals or in a group discussion.
- ❑ Create a class list of debugging strategies by collecting students' problem finding and problem solving approaches.

RESOURCES

- ❑ Unit 4 Debug It! handout
- ❑ Unit 4 Debug It! studio
  http://scratch.mit.edu/studios/475634

REFLECTION PROMPTS

+ What was the problem?
+ How did you identify the problem?
+ How did you fix the problem?
+ Did others have alternative approaches to fixing the problem?

REVIEWING STUDENT WORK

+ Were students able to solve all five bugs? If not, how might you clarify the concepts expressed in the unsolved programs?
+ What different testing and debugging strategies did students employ?

NOTES

+ This activity provides an opportunity to check in with students who might need some additional attention or support, particularly around the concepts of conditionals (e.g., if), operators (e.g., arithmetic, logical), and data (e.g., variables, lists).

NOTES TO SELF

- ❑ .....
- ❑ .....
- ❑ .....
- ❑ .....