# 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

❏ _____
❏ _____
❏ _____
❏ _____

# DEBUG IT!

HELP! CAN YOU DEBUG THESE FIVE SCRATCH PROGRAMS?

In this activity, you will investigate what is going awry and find a solution for each of the five Debug It! challenges.

## START HERE

- ☐ Go to the Unit 1 Debug It! studio: http://scratch.mit.edu/studios/475483
- ☐ Test and debug each of the five debugging challenges in the studio.
- ☐ Write down your solution or remix the buggy program with your solution.

## FEELING STUCK?

THAT'S OKAY! TRY THESE THINGS...

☐ **DEBUG IT! 1.1** http://scratch.mit.edu/projects/10437040

When the green flag is clicked, both Gobo and Scratch Cat should start dancing. But only Scratch Cat starts Dancing! How do we fix the program?

☐ **DEBUG IT! 1.2** http://scratch.mit.edu/projects/10437249

In this project, when the green flag is clicked, the Scratch Cat should start on the left side of the stage, say something about being on the left side, glide to the right side of the stage, and say something about being on the right side. It works the first time the green flag is clicked, but not again. How do we fix the program?

☐ **DEBUG IT! 1.3** http://scratch.mit.edu/projects/10437366

The Scratch Cat should do a flip when the space key is pressed. But when the space key is pressed, nothing happens! How do we fix the program?

☐ **DEBUG IT! 1.4** http://scratch.mit.edu/projects/10437439

In this project, the Scratch Cat should pace back and forth across the stage, when it is clicked. But the Scratch Cat is flipping out – and is walking upside down! How do we fix the program?

☐ **DEBUG IT! 1.5** http://scratch.mit.edu/projects/10437476

In this project, when the green flag is clicked, the Scratch Cat should saw 'Meow, meow, meow!' in a speech bubble and as a sound. But the speech bubble happens before the sound – and the Scratch Cat only makes one 'Meow' sound! How do we fix the program?

## FINISHED?

- ☐ Make a list of possible bugs in the program.
- ☐ Keep track of your work! This can be a useful reminder of what you have already tried and point you toward what to try next.
- ☐ Share and compare your problem finding and problem solving approaches with a neighbor until you find something that works for you!

- + Discuss your testing and debugging practices with a partner. Make note of the similarities and differences in your strategies.
- + Add code commentary by right clicking on blocks in your scripts. This can help others understand different parts of your program!
- + Help a neighbor!