

girls who
code

Girls Who Code At Home

Brave, Not Perfect
Scratch Debugging

Activity Overview

In this activity we will dive deeper into what it means to be **brave, not perfect** when coding! Just because a program may not work as expected doesn't mean that it is a failure. There are many skills that we can learn and apply to fix mistakes and improve our products or projects.

Debugging is a strategy that computer scientists use to find and fix problems, or **bugs**, in their program. In this activity you will help to debug, or fix, three programs in Scratch! Before diving into this activity we recommend reading about the featured Woman in Tech, Ayanna Howard. Ayanna's first project with NASA was to create a robot that learns about the environment on Mars. Learn more about how Ayanna uses courage and confidence to innovate.

Materials

- [Online Scratch](#) or [Offline Scratch](#)
- [Debugging Challenge 1](#)
- [Debugging Challenge 2](#)
- [Debugging Challenge 3](#)
- Optional: Debugging Challenge Handout
- Debugging Challenge Handout Solutions

Women in Tech Spotlight: Ayanna Howard



Image Source: [Black Sci-Fi](#)

Role models come in all shapes and sizes. In fact, Dr. Ayanna Howard's first role model wasn't even human! Bionic Woman, a robotic superheroine, inspired Dr. Howard's passion for building robots and engineering at an early age. She went on to study electrical engineering and computer science in college and graduate school, and eventually earned a business degree.

Continuing her love for learning, robotics, and electrical engineering, Dr. Howard currently serves as a professor of bioengineering and the co-founder and Chief Technology Officer of the educational robotics company Zyrobotics. On top of this, Dr. Howard has developed robots that are learning to inhabit Mars with NASA.

Watch this [video](#) to learn more about Dr. Howard's first job at NASA. She also discusses why diversity in the workplace is important, and how she stood up to an uncomfortable situation at work.

Reflect

There's more to being a computer scientist than just being great at coding. Take some time to reflect on how Ayanna and her work relates to the strengths that great computer scientists focus on building - bravery, resilience, creativity, and purpose.



Dr. Howard showed bravery when she spoke up in an uncomfortable situation at work. What would you say to yourself in that moment?

Share your responses with a family member or friend. Encourage others to read more about Ayanna to join in the discussion!

Step 1: What is Debugging? (5 mins)

Think of a time you tried to solve a problem, this could be from a difficult assignment at school, or trying to locate a missing item. Were you able to solve it right away? Often when we try to solve a problem, we might experience several failures before we succeed. Programmers spend most of their days trying to find and solve problems in their code! Failure is a big part of computer science 😊 .

An error in a computer program or piece of hardware is called a **bug**. The process of identifying and removing errors or bugs from computer hardware or software is called **debugging**. The origins of these terms stem back to Grace Hopper, one of the early pioneers of computer science. While working on one of the first computers, Grace Hopper's team found a moth inside the computer which caused an error, an actual physical bug! Grace's journal entry of the taped moth is now referred to as the first recorded computer bug in history. It is on display at the Smithsonian Museum of American History in Washington D.C.

The first recorded computer bug

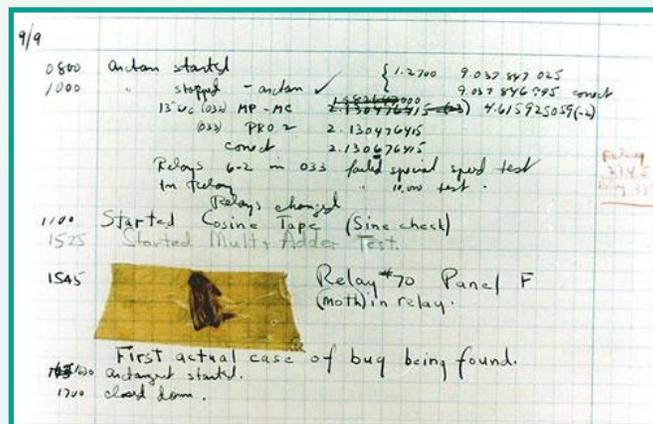


Image Source: [Atlas Obscura](#)

Take some time to think what the phrase **Brave, Not Perfect** means to you. Why it is important to be **brave** when trying something new instead of trying to be **perfect**? Oftentimes we will face challenges when trying to learn something new. It is important to be **resilient** and to try to look at our mistakes and challenges as learning opportunities. Debugging is an opportunity to learn from your mistakes. These strategies often help you tackle larger and more complicated challenges in the future.

Step 2: Sign Into Scratch & Navigate the Interface (5-10 mins)

Scratch is a free programming platform and block-based programming language developed by MIT that allows you to program interactive stories, games, animations.

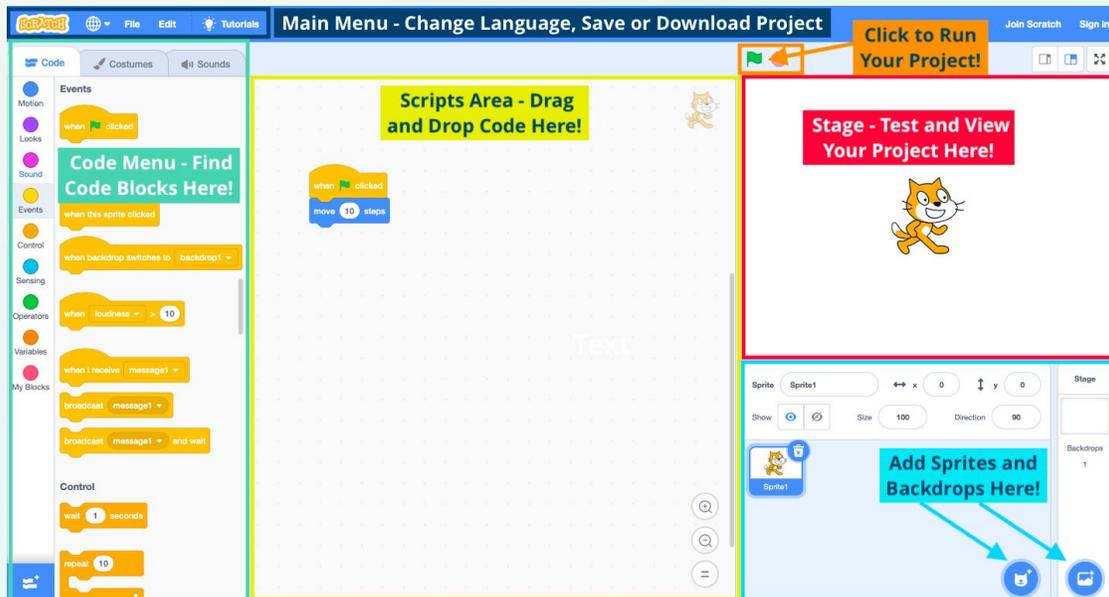
1. Sign up or login into [Scratch](#).

In order to save your work on Scratch's online platform you'll need to create an account if you don't already have one. Follow the instructions on the sign up form to create an account. If you are under 13 you'll need your parent's email address to sign up. If you don't want to create an account you can also download and use the [offline version of Scratch 3.0](#).

2. Explore the Scratch interface.

If you are new to Scratch take a few minutes to **Create**   a new project so you can explore the Scratch interface. You can also watch this [Getting Started](#) tutorial from Scratch!

In the image below are some of the key sections on the Scratch platform that you will be interacting with.



Step 3: Review Scratch Debugging Strategies (5 mins)

Before you start debugging, take a few minutes to review some suggested strategies you could use when trying to find and fix a bug in your program.

1. **Describe the error (or bug).** How did you notice there was a bug in the program? Think about what you expected to happen and what actually happened.
2. **Read through your code and think of possible errors.** Read through each line of code line by line. You can even say each line of code out loud! While you are reading back through your code, think about which line(s) of code (or blocks for Scratch) may be the source of the error or unexpected results.
3. **Place code markers throughout your code as check points.** If your code is long it can be helpful to divide your code into multiple sections. In Scratch you can use a **say** block so that you know where your program is in your line of code. This strategy can help narrow down the possible locations of the bug.



For example, if you know that when your first marker is hit (meaning the sprite says the text in the say block you are using as a marker) and the program is still working as expected, then you know that the bug does not likely exist before the first marker. If you notice the bug in your program before you hit the second marker, you know the bug is likely located in the code between the first and second marker.

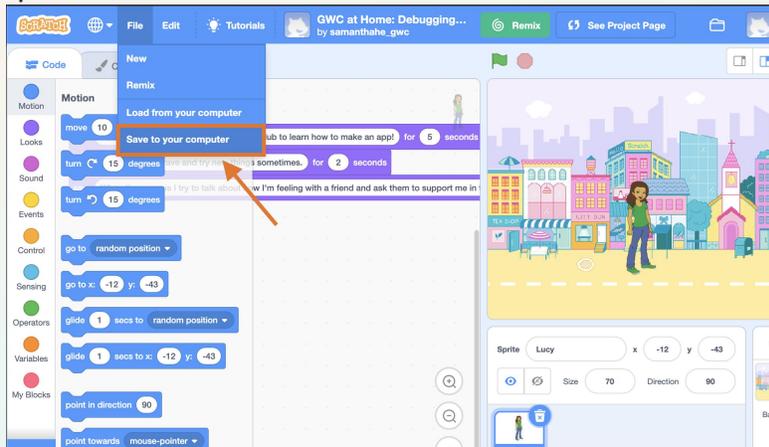
4. **Once you have found the bug, try to understand why the code is a bug.** Think of why the line of code or code block causes an error in your program. Some common errors might include a spelling error, placing the code blocks in the wrong order, or using the wrong block. Some code editors include error messages that describe why a particular line of code is a bug. Scratch does not have these error messages, so you'll have to do more of the thinking!
5. **Fix and Test your code.** Try fixing the line of code and then running your program again. Does the bug still exist? If so, try out a different solution and test again! Does a bug happen in a different area? Then maybe you had a second bug in the program, go through the steps again to find your new bug. If there are no bugs in your program then that means you fixed your bug!

Step 4: Tackle Debugging Challenge 1 (5-10 mins)

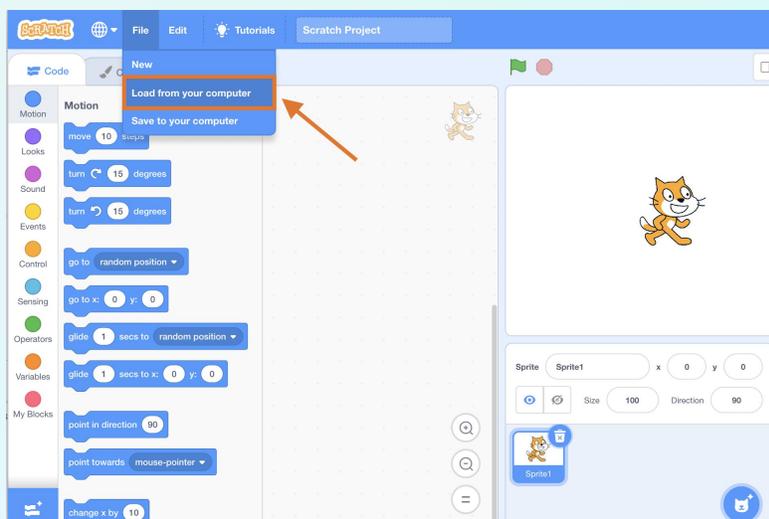
Prior Knowledge: In order to solve this challenge you should be familiar with [Event](#) and [Looks](#) blocks. Try watching this [Getting Started](#) tutorial to familiarize yourself with these blocks.



1. **Remix the [starter code](#).** Click the  button on the top right to make a copy of the project.
 - If you are using Scratch Offline Editor you will need to save a copy of the project to your computer. Click the  button to view the project code.
 - Click **File** at the top navigation bar and choose **Save to your computer** option in the Drop-down menu.



- Open Scratch Offline Editor on your computer. Click **File** at the top navigation bar and choose **Load to your computer** option in the Drop-down menu. Find your saved project file on your computer and click **Open**.



Step 4: Tackle Debugging Challenge 1 (Continued)

2. **Test the Program and Identify the Bug.** In this project Lucy, our main sprite (or object), introduces herself to the user. However, when the green flag is clicked nothing happens! Run the program by pressing the Green Flag . Review the code and see if you can identify the bug. Use the **challenge handout** on pages 12-13 to help solve and reflect on the bug.
3. **Fix the Bug and Check your Solution [here](#).** See on page 14 for a more in depth discussion of this bug!
4. **Reflect on your debugging process.** Think about what the bug was and how it affected the program. When learning how to code, many programmers keep a cheat sheet with common errors to help them learn from their mistakes when coding other projects later.

Step 5: Tackle Debugging Challenge 2 (5-10 mins)

Prior Knowledge: In order to solve this challenge you should be familiar with [Event](#), [Looks](#), [Motion](#), [Sensing](#) and [Loops](#) blocks. Try watching this [Code a Cartoon](#) tutorial to familiarize yourself with these blocks.



- 1. Remix the [starter code](#).** Click the  button on the top right to make a copy of the project.
 - If you are using Scratch Offline Editor you will need to save a copy of the project to your computer. Click the  button to view the project code.
 - Click **File** at the top navigation bar and choose **Save to your computer** option in the Drop-down menu.
 - Open Scratch Offline Editor on your computer. Click **File** at the top navigation bar and choose **Load to your computer** option in the Drop-down menu. Find your saved project file on your computer and click **Open**.
- 2. Test the Program and Identify the Bug.** In this program Avery should start in the middle and introduce herself, she then walks towards the right until she reaches the edge to go to the coding club. It works the first time we press the Green Flag  but when we press the green flag again Avery is still on the right side instead of starting in the middle of the screen. Run the program by pressing the Green Flag. Review the code and see if you can identify the bug. Use the **challenge handout** on pages 12-13 to help solve and reflect on the bug.
- 3. Fix the Bugs and Check your Solution [here](#).** Note, you may have coded Avery to start at a different position than what is listed in the solutions. See on page 15 for a more in depth discussion of this bug!
- 4. Reflect on your debugging process.** Think about what the bug was and how it affected the program. When learning how to code, many programmers keep a cheat sheet with common errors to help them learn from their mistakes when coding other projects later.

Step 6: Tackle Debugging Challenge 3 (5-15 mins)

Prior Knowledge: In order to solve this challenge you should have a knowledge of [Event](#), [Looks](#), [Sounds](#), [Wait](#), and [Loops](#) blocks. Try watching this [Code a Cartoon](#) tutorial to familiarize yourself with these blocks.



- 1. Remix the [starter code](#).** Click the  button on the top right to make a copy of the project.
 - If you are using Scratch Offline Editor you will need to save a copy of the project to your computer. Click the  button to view the project code.
 - Click **File** at the top navigation bar and choose **Save to your computer** option in the Drop-down menu.
 - Open Scratch Offline Editor on your computer. Click **File** at the top navigation bar and choose **Load to your computer** option in the Drop-down menu. Find your saved project file on your computer and click **Open**.
- 2. Test the Program and Identify the Bug.** When you press the Green Flag  Ada should tell you about herself and then dance to the music. When Ada starts dancing there is something wrong with the music! It keeps restarting over and over again every time Ada dances. How can we fix the code so that the music plays all the way through while Ada dances? Run the program by pressing the Green Flag. Review the code and see if you can identify the bug. Use the **challenge handout** on pages 12-13 to help solve and reflect on the bug.
- 3. Fix the Bugs and Check your Solution [here](#).** Note, you may have coded Avery to start at a different position than what is listed in the solutions. See on page 16 for a more in depth discussion of this bug!
- 4. Reflect on your debugging process.** Think about what the bug was and how it affected the program. When learning how to code, many programmers keep a cheat sheet with common errors to help them learn from their mistakes when coding other projects later.

Step 7: Share Your Creation (5 mins)

1. **Share your project on Scratch.**
Once you have solved a challenge press the Share button in Scratch. Add in the instructions section how you solved the bug!
2. **Share how you are tackling challenges with Girls Who Code at Home!**
Don't forget to share your projects on social media. Tag @girlswhocode #codefromhome and we might even feature you on our account!

Debugging Challenge Handout

Instructions: As you go through each of the challenges think about the following questions.

- What is the bug? What was supposed to happen? Are there multiple bugs?
- How did the bug affect the program and why?
- How did you fix the bug?

Debugging Challenge 1 (5-10 mins)

Starter Code: <https://scratch.mit.edu/projects/387548698/>

Bug(s):

How it Affected the Program and Why:

How did you fix the bug(s):

Debugging Challenge Handout

Debugging Challenge 2 (5- 10 mins)

Starter Code: <https://scratch.mit.edu/projects/387549618/>

Bug(s):

How it Affected the Program and Why:

How did you fix the bug(s):

Debugging Challenge 3 (5-15 mins)

Starter Code: <https://scratch.mit.edu/projects/387851932/>

Bug(s):

How it Affected the Program and Why:

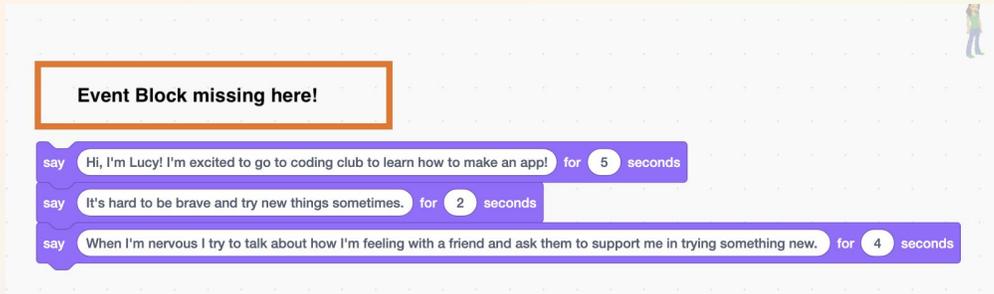
How did you fix the bug(s):

Debugging Challenge 1 Solution

Bugged Starter Code: <https://scratch.mit.edu/projects/387548698/>

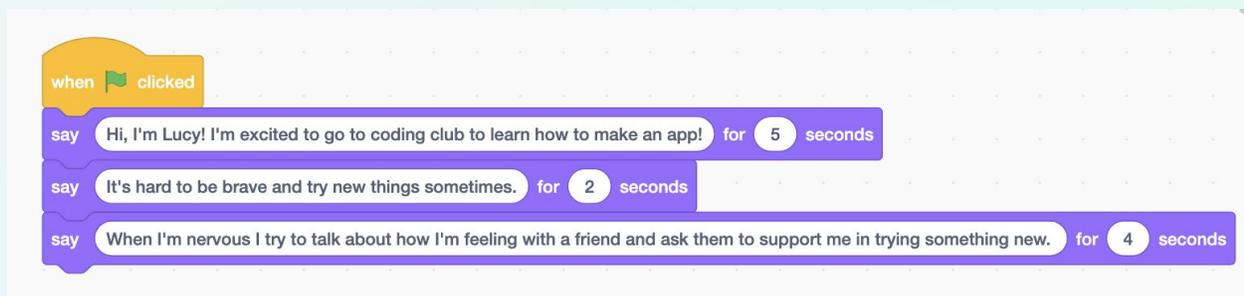
Sample Solution Code: <https://scratch.mit.edu/projects/388218006/>

Bug(s): When clicking the green flag  Lucy should say her introduction. Instead, nothing happens! This was because there was no event block present in the code.



How it Affected the Program and Why: Without an event block the computer doesn't know when to run the code we have written. Therefore when the green flag is clicked nothing happened because we didn't tell the computer to do anything.

How did you fix the bug(s): We must tell the computer that **When the green flag is clicked** do the code that we have written. We used the  block and attached this at the front of the other code blocks.

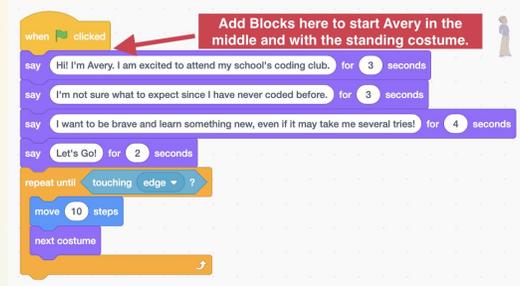


Debugging Challenge 2 Solution

Bugged Starter Code: <https://scratch.mit.edu/projects/387549618/>

Sample Solution Code: <https://scratch.mit.edu/projects/388333942/>

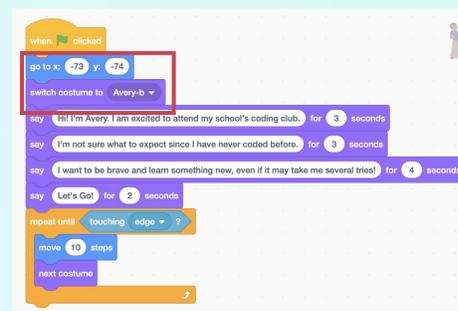
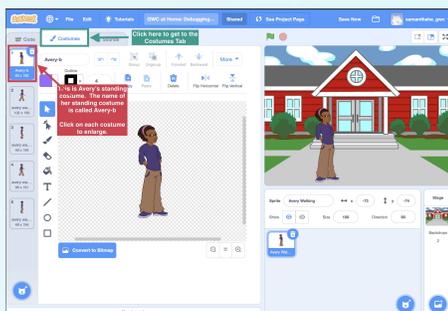
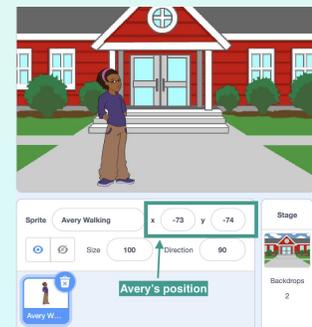
Bug(s): This program contains **two** bugs. After clicking the Green Flag a second time, Avery no longer starts at the same starting spot in the middle of the screen. Instead she begins where she last left off at the right of the screen and in a walking costume. The first bug is her position and the second bug is her costume (or look).



How it Affected the Program and Why: In this program Avery should start in the middle and introduce herself, she then walks towards the right until she reaches the edge to go to the coding club. Because we don't have code that always starts Avery at the same position every time the Green Flag is clicked, Avery begins her sequence of code where she last left off. In our case, at the edge of the screen on the right.

How did you fix the bug(s): In order to have Avery always starting at the same location we must use the `go to x: -73 y: -74`. The easiest way to pick Avery's starting position is to use your mouse and drag Avery to where you want her to start. You should notice that the numbers after x: and y: change according to Avery's position. You should also see this in the sprite description below the Scratch stage. We use the go to block as the very first block after our Event block `when clicked`.

Next, we want Avery to start in the costume that shows her standing. For this we need to use the `switch costume to Avery-b` right after we set her starting position to make sure she always starts in the standing costume. If you are not sure of which costume is her standing position, look in the **Costumes** tab and record the name of the costume you want her to start in. We code these two new blocks first because we want to set our sprite at the same location right when the green flag is clicked before Avery starts talking and walking!

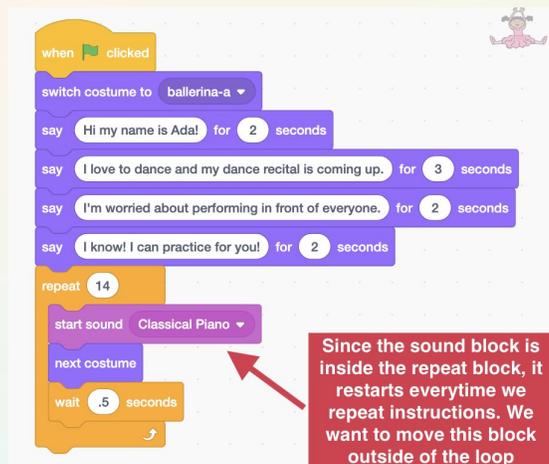


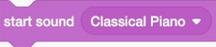
Debugging Challenge 3 Solution

Bugged Starter Code: <https://scratch.mit.edu/projects/387851932/>

Sample Solution Code: <https://scratch.mit.edu/projects/388346112/>

Bug(s): When you press the Green Flag , Ada should tell you about herself and then dance to the music. When Ada starts dancing there is something wrong with the music! It keeps restarting over and over again every time Ada moves. The bug has to do with the sequence, or order, that the blocks are coded.



How it Affected the Program and Why: Notice that our sound block  is coded inside the repeat loop. This means that every time we loop through the code inside, the Classical Piano restarts again!

How did you fix the bug(s): We need to pull our sound block outside of the loop but where? We need to think back on what we want to happen. Try experimenting moving the sound block before and after the repeat block. If we move the sound block after the repeat loop the music plays after Ada dances, this is not what we want. If we move the sound block before the repeat loop, first the music plays and then Ada dances. This is what we want! You can decide if you want the sound to start when she dances or right in the beginning as she makes her introduction.

