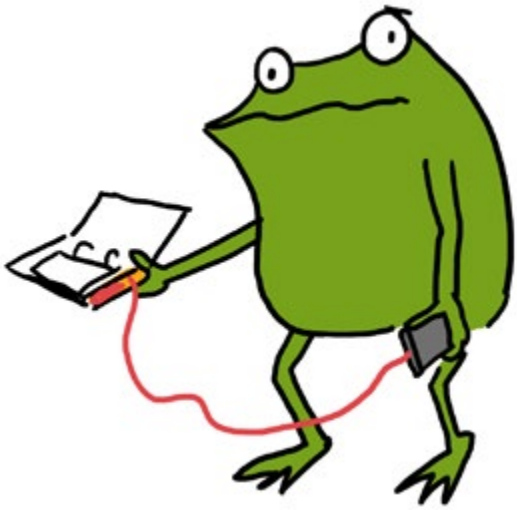
Debugging



Stuff isn't working? No matter! That's what debugging is for! **Debugging** means looking closely at our project, finding the problems — also known as **bugs** — and then fixing them so that our project works as expected.

WELCOME TO MY WORLD! DON'T WORRY WHEN THINGS DON'T WORK THE FIRST TIME. FIGURING OUT WHAT WENT WRONG IS HOW DISCOVERIES ARE MADE!

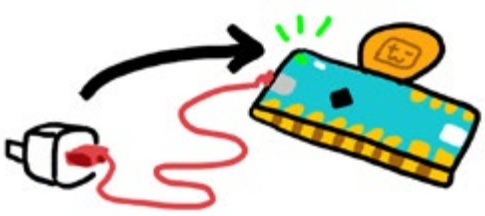
BREAKING IT DOWN

When a project doesn't work, it can seem overwhelming. The problem could be anywhere!

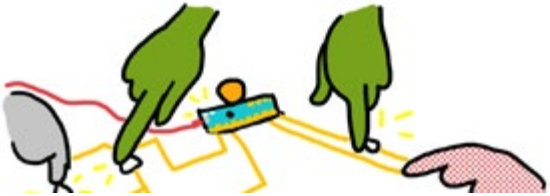
That's why we break it down into smaller parts and look at them one by one. Then it's not so scary anymore! It's just like how we take small bites when we eat, rather than swallowing the whole meal all at once!

Here are some of the smaller pieces we can break our Love to Code project into:

Power:
Is the Chibi Chip getting enough electricity to turn on and run our circuit?



Circuit:
Is everything in our circuit connected properly?

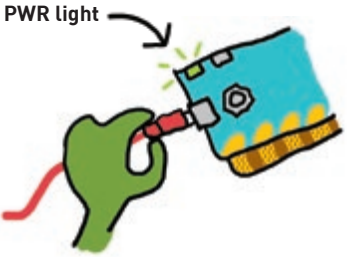


Upload:
Are we able to send code from our programming device to the Chibi Chip?

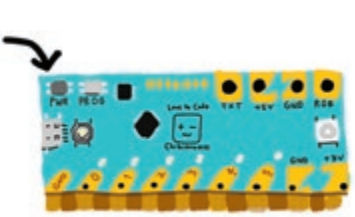
Code:
Are there errors in our code? Does our code actually say what we mean?

POWER

Let's start by checking the power! If everything is working properly, the power (PWR) light on the Chibi Chip will be green to show that it has enough power.



1. Is the PWR light off? If so, it means the Chibi Chip is off because it isn't getting enough power! Try plugging the Chibi Chip into a power supply that you've recently used to charge a phone, like a USB wall plug or computer's USB port. Phones take a lot of power to charge, so if the USB port can charge a phone, it can power a Chibi Chip!

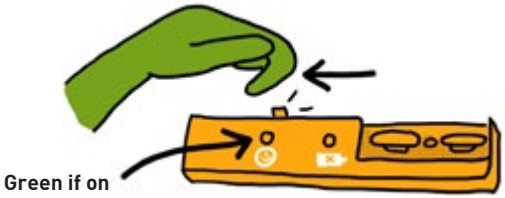


2. Is the PWR light red instead of green? That typically means we have a short circuit connecting +3V and GND directly to each other, draining power from the Chibi Chip! If you're powering the Chibi Chip from a computer, an error message may also pop up on your screen about too much power or current being drawn.

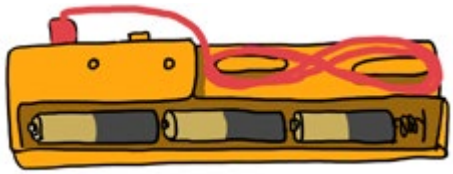
If this happens, unplug the Chibi Chip from the power source, find the connection causing the short circuit, and remove it. To learn more about short circuits, go to page **1-8**.

If you're using a Chibi Book for power, it's worth checking to see if the battery pack is working properly. Is the battery pack's green power light on? If not, here are a few things to check:

3a. Is the battery pack switched on? Make sure the switch at the top of the battery pack is flipped to the left, and that the green light is on.



3b. Are the batteries inserted properly? There should be three AA batteries placed in the battery pack like this with the + side (nub side) facing left:



3c. Is the red "low battery" light on or flashing? Is the green light flashing on and off? This usually means the batteries are running really low. The Chibi Book is trying to find the power to run your circuit, but exhausting itself and shutting down, only to try again after a short rest. Try placing fresh batteries into the Chibi Book's battery pack. The Chibi Book works best with alkaline or NiMH rechargeable batteries!

ONCE WE KNOW POWER IS GETTING TO THE CHIBI CHIP, WE CAN CHECK THE REST OF OUR CIRCUIT FOR OTHER PROBLEMS!

CIRCUIT

Is the Chibi Chip powered on, but the LEDs are still not turning on as expected? There might be a bug in the connections of our circuit! Maybe the circuit is incomplete because we forgot to make a connection, or the connection is faulty. Or maybe something is connected that should not be, causing a short circuit.

LET'S TAKE A LOOK AT SOME COMMON CIRCUIT BUGS!

1. Are your Chibi Light LEDs installed in the correct direction? Make sure that the pointy (-) end of every LED is connected to GND, and the wide (+) side of every LED is connected to a numbered pin or +3V. Otherwise, the LED is installed backwards, and it won't turn on.

YAY!

NOPE.

2. Are any LEDs causing a short circuit? Make sure that the shiny metal pads of your LEDs are touching only one strip of copper tape. If one pad of an LED is touching two different copper wires, then there is a short circuit and your light will not turn on. In the example below on the right, the wide (+) side is accidentally touching both GND and +3V, causing a short circuit!

YAY!

NOPE.

3. Is there enough overlap between the pads of your LED and the copper tape? There needs to be plenty of overlap between the metal pad of your LED and the copper tape for there to be a strong electrical connection. If the overlapping area is too small, then power can not flow to the LED.

YAY!

NOPE.

4. Are all the LEDs stuck down firmly? The conductive adhesive used on circuit stickers require a good press to make a solid connection. So, try pressing down on the metal pads of your LEDs to make the connection stronger. If an LED turns on when pressed, and off after letting go, that means there's a weak connection between the copper tape and the LED's metal pads.

If you have more than one LED connected in parallel, and only some of them are on, the most likely issue is a weak connection on the LEDs that aren't turning on. You can fix these weak connections by taping conductive fabric patches between the LED's metal pads and the underlying copper tape.

YAY!

NOPE.

5. If you've tried everything and an LED is still not turning on, try switching it out for a new LED. LEDs can break if they are creased, or if they are exposed to water, dirt, or grease.

CIRCUIT (CONT'D)

6. Is the Chibi Chip aligned properly with the circuit? Make sure the shiny metal pads on your Chibi Chip line up with and touch the copper tape of your circuit.

YAY!

NOPE.

7. Is the copper tape really bumpy or wrinkly? If so, sometimes bumps and wrinkles can prevent solid connections to your LEDs or Chibi Chip. If this is the case, try smoothing out the tape by rolling over it with the flat side of a pen or pencil.

YAY!

NOPE.

8. Is the copper tape or LED sticker not sticky anymore? Make sure your hands are clean and dry before working with the copper tape and stickers. If the stickers or tape get dirty, they may lose their tack, causing weak connections in the circuit. If this happens, try patching weak connections with conductive fabric patches.

9. Is there a tear in the copper tape? If so, the tear breaks the circuit's connection. Fix tears and restore the connection by taping over both sides of a tear with a single conductive fabric patch.

10. Is your circuit connected at turns and corners? Copper tape often tears at turns, and it is not enough to stick two pieces of copper tape on top of each other. When making a corner using two pieces of copper tape, reinforce the connection using a conductive fabric patch after sticking down the second piece of copper tape.

YAY!

NOPE.

Make reliable turns with a single piece of copper tape and save on conductive fabric patches using this origami trick:

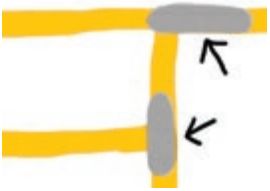
1. Fold tape back, so the sticky side faces up.

2. Flip and turn the tape at the same time.

3. Flatten the corner, and you're done!

CIRCUIT (CONT'D)

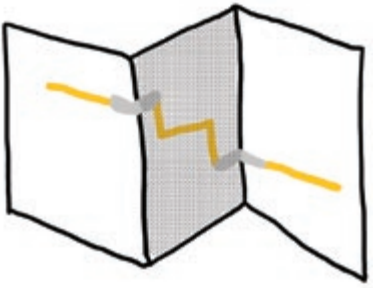
11. Are there branches in your circuit, or do you need to extend your copper tape with another piece? Make sure to use a conductive fabric patch to connect multiple pieces of copper tape. Just sticking two pieces of copper tape on top of each other will not create a strong or reliable electrical connection. Even if the circuit seems to works at first, over time the connection will break down.



YAY!

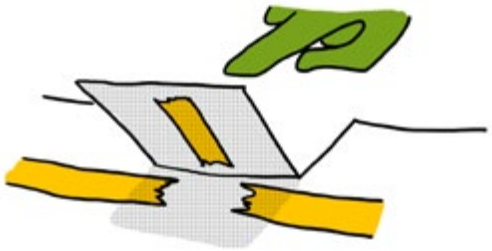
NOPE.

12. Do you have a moving hinge in your circuit? Make sure to reinforce it with a conductive fabric patch. Tears are especially common at places where the copper tape gets folded repeatedly. Copper tape will crack when folded too many times.



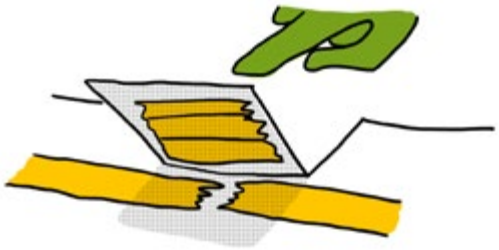
FABRIC PATCHES WON'T DEVELOP CRACKS EVEN WITH REPEATED FOLDING, SO THEY'RE GREAT FOR REINFORCING MOVING OR BENDING PARTS OF A CIRCUIT!

13. Is a switch not responding when pressed? If so, make sure that the copper tape on the switch actually lines up with the gap in the circuit. If it's not aligned, the copper tape won't close the circuit when the switch is pressed, and the switch will not work.



NOPE.

Make the copper tape patch large and the circuit gap small. This makes it easier for your switch to close the circuit. This is especially important for more advanced switches, such as the wind sensing switch (page **3-28**), where the switch isn't actually pressed by hand.



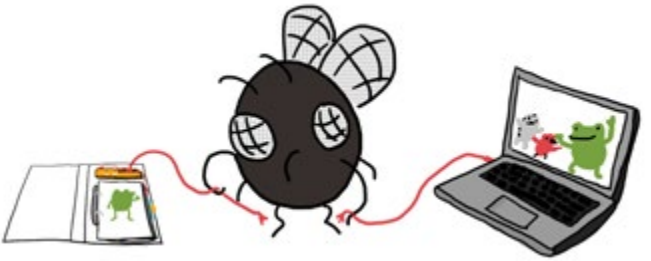
YAY!

14. Is the switch sometimes working and sometimes not when pressed? That means the switch connection is unreliable. Check that the copper tape is smooth and clean on both halves of the switch. Then, try ways to increase the pressure between the pieces of copper tape in your switch.

For example, in the pocket character switch (page **3-31**), if you use a thicker paper, this will make the switch stiffer and more springy. This applies more pressure to the connection, helping to make your switch more reliable.

UPLOAD

Even if the circuit is done, the Chibi Chip needs to be programmed correctly for our project to work. Sometimes there are problems when we try to send code from our programming device to the Chibi Chip. Here's how to check if this is an issue!



We test the upload process by trying to upload the Blink example program. Save any code you've written and open up the blink example code by selecting **Examples → Love to Code Vol 1 → Basic Blink**. We start with this known code because it's easy to tell if it's working properly. If the upload is successful, we will see pin 0 blink!

If the Basic Blink didn't upload, let's check out some possible reasons why:

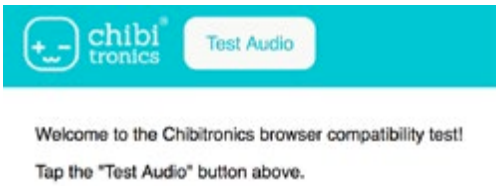
1. Is the volume turned up all the way up? Is the sound acccidentally muted? Make sure to unmute your sound and turn the volume all the way up so that the Chibi Chip can hear the code. One of the most common upload problems is that the audio is simply too quiet!

2. Is the code being converted into sound? Click the "Upload" button. Within a few seconds, this orange "sound banner" should appear at the bottom of the screen:

If the sound banner does not appear, the code is not being translated into sound. If this happens, try refreshing the browser. Then, re-open the blink example code, and click "Upload" again.

3. Can your browser play web audio? Try unplugging the audio cable from your programming device for a moment, turning the volume up to its maximum level, and clicking upload. When the sound banner appears, you should hear a staticky sound. This staticky sound is the code that the Chibi Chip is listening for.

If you can't hear the code playing, try testing to see if your browser is compatible with the Love to Code system. To run the test, visit **ltc.chibitronics.com/test** and click on the "Test Audio" button. If you hear a short tune play, that means your browser is compatible. If not, try switching to an up-to-date version of Chrome, Firefox or Edge.



UPLOAD (CONT'D)

4. Is the audio being distorted? Some laptops automatically apply audio "enhancements" (such as Dolby Audio or bass boost). These enhancements will distort sound in a way that the Chibi Chip may not be able to understand. If you have a Windows computer, particularly those made by Lenovo, try following these instructions to disable pre-loaded audio distortions:

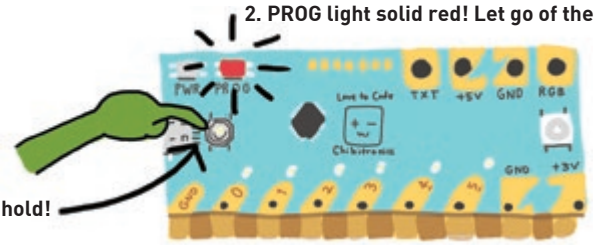
2. Select "Turn off Dolby Audio" (you can turn it on again using the same menu item)

1. Right-click "Dolby" icon

5. Do you hear a static sound while programming the Chibi Chip? That means your audio cable isn't plugged all the way in. Make sure to push the audio cable all the way into your programming device, so that the Chibi Chip is hearing the code, and not you!

6. Is the Chibi Chip in program mode? Before clicking upload, make sure to press and hold the programming (PROG) button on the Chibi Chip until the PROG LED blinks and stays red. Otherwise the Chibi Chip won't know to listen for new code.

2. PROG light solid red! Let go of the button now.



1. Press hard & hold!

7. Is the Chibi Chip hearing the audio code? During upload, the red PROG light on the Chibi Chip should blink to show that it hears the code. If the PROG light stays solid red, even though the sound should be playing, it means the audio is not making it to your Chibi Chip. If this is the case, try turning up the volume on your device and check that sounds are playing (**Step 3**).

8. Does the PROG light turn green? If so, that means the upload is successful. If your project still isn't behaving as expected, try checking the circuit or the code!

If the PROG light just blinks red, but never turns green that means your Chibi Chip heard some of the sounds, but was not able to hear the entire program. Check again that your volume is turned all the way up (**Step 1**), and check for hidden add-ons or plug-ins that could be distorting the audio (**Step 4**). Furthermore, if another kind of device works, this means there is likely some kind of distortion in your programming device's audio path.

8. Still stumped? Send us a note at **help@chibitronics.com** and we'll try our best to debug with you!

CODE

Sometimes there will be errors in our code that makes our circuits do something other than what we intended. In this case, we have to debug our code!

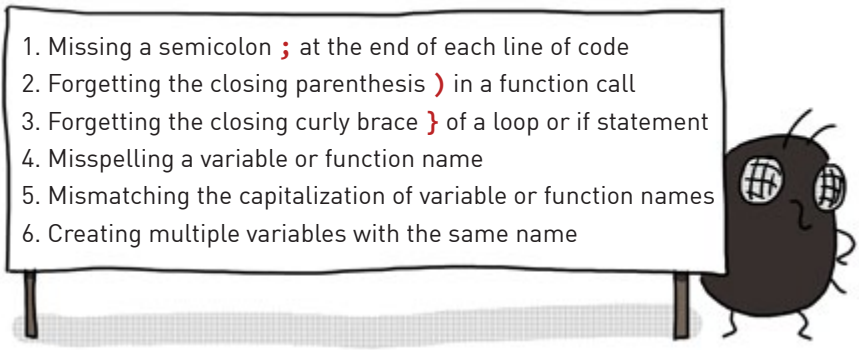
DEBUGGING CODE CAN OFTEN BE A LONG AND FRUSTRATING PROCESS, BUT REST ASSURED, IT'S REALLY SATISFYING WHEN YOU FINALLY FIGURE OUT WHAT'S WRONG AND GET YOUR PROJECTS WORKING!

Clicked "Upload" on the browser, but the sound bar doesn't appear?
There may be formatting errors in your code. If you're able to upload the blink example code but not your own code, this is likely the case.

The code editor needs your code to be written in exactly the right format, otherwise it wont understand the code and cannot **compile** it. Compiling means translating the text code in your browser into the code song that a Chibi Chip can understand.

As a result, little errors in how the code is written, called **syntax errors**, will stop an entire program from uploading!

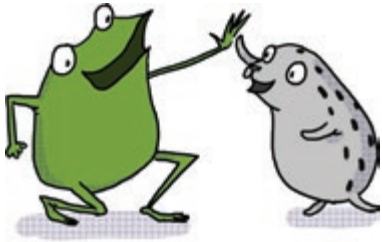
Here are some common errors to check for:



Luckily, for common syntax errors like these, the editor will add a small red circle next to, or just after, the code lines with problems. If you hover over these circles with your mouse, the editor will pop up additional information about the error. This way it's easier to find the mistakes and fix them!

```
1  //Love to Code
2
3  int LED = 0;
4  int LED = 1;
5
6  void setup() {
7    outputmode(LED);
8  }
9
10 void loop() {
11   on(LED);
12   pause(1000;
13   off(LEd)
14   pause(1000);
15
```

Can you spot all the errors in the code above? How would you fix them?



CODE (CONT'D)

Code uploaded properly, but not behaving as intended? That means that the code is formatted correctly, so it compiled and uploaded, but there is an error in what the code tells the circuit to do, causing the program to behave in an unintended way. This type of error is called a **logic error**.

Logic errors can be challenging to spot and fix because we have to figure out our error based on the unexpected behavior of our circuit.

Some logic errors can be quick to spot. Here are a few examples of common logic errors:

1. Are there enough delays? If there are not enough delays in your program, or if the delays are not long enough, the logic might be running as intended, but the effects are happening too quickly for us to see.

2. Do the pins in the code match the pins in the circuit? It's easy to forget which pin was used when crafting a circuit. For example, when adding a switch, make sure to update the pin number in the example code to match the pin number you've wired the switch to!

3. Are the LEDs blinking on and off, but not very brightly? It may be because the LED pin is still in input mode, which is the default mode. If this is the case, the LED will still blink, but the pin will not be able to turn an LED fully on. Make sure to set every pin meant to turn on an LED to be an output using `outputMode(pin number);` in the setup code.

CODE (CONT'D)

Most logic errors are hard to spot. But don't worry: finding bugs and fixing them is all part of learning to code! Here are some tips for finding the trickier bugs:

1. Pretend to be the Chibi Chip, and trace through the code line by line. Tracing through a program slowly can help catch many bugs. For example: "turn light on", "wait 1 second", "turn light off", "loop ends, repeat", "turn light on" - Aha, I was missing a delay after the "light off!", so the light turned off and on so quickly I couldn't see it!

2. Test one change at a time. If you make several changes at once, you may not know which change actually fixes the problem. Also, sometimes changes can introduce new bugs, so even if one change fixed the bug, the other change could have broken it again!

3. Reduce your code to the simplest possible version to isolate a bug. Just as we broke our Love to Code projects into power, circuit, upload and code portions to simplify debugging, we can break a program down into smaller pieces of code so it's easier to work with.

For example, if you have a program that is blinking five lights, but one of those lights is not working, save your program into a new file, and simplify it so that it controls only the one light that's having problems.

Once you figure out what's going on with the simple code for one light, you can refer to your original version and apply the fixes, testing each change as you go.

CODE (CONT'D)

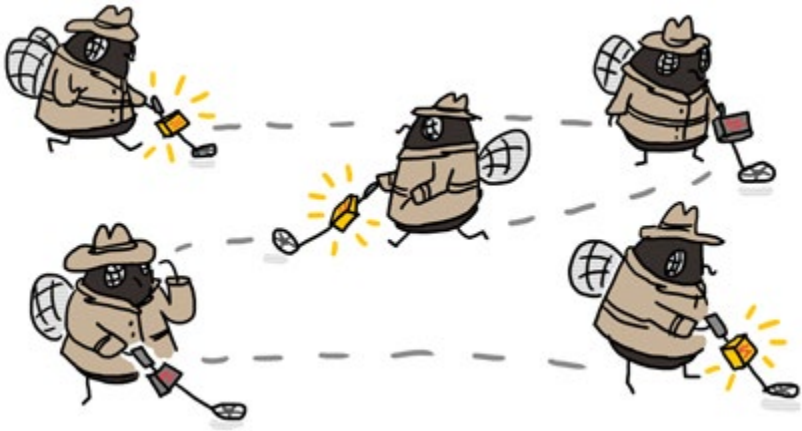
A useful tool for finding and fixing bugs is to add a little extra code in your program that helps monitor the Chibi Chip's progress in running your code. For example, we could insert a few lines of code that turn an LED on and off at a specific point in a program.

If the LED blinks, that means the Chibi Chip is able to run up to that part of the code. Likewise, if the LED doesn't blink, it means that the code leading up to the blink isn't being run. This way, we can use the blinking LED as an indicator for tracing through our code. If possible, try to use an LED that you aren't already using in your project! Below is a starting point for a blinkometer that you might find handy:

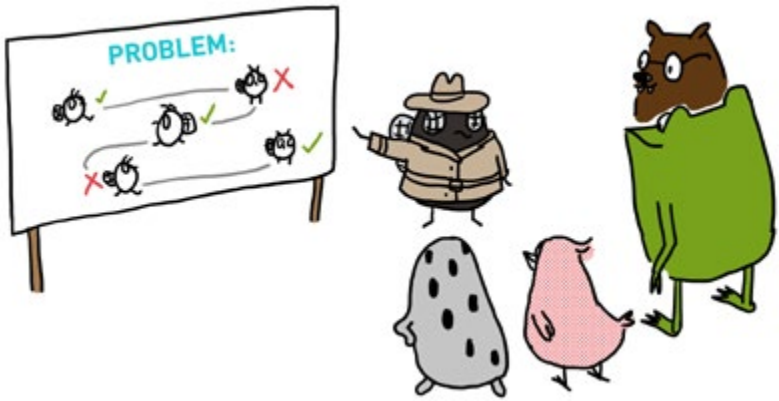
```
// these five lines of code are a blinkometer you can insert
// in your program to see how far it has run!
  outputMode(4); // ensure pin 4 can drive an LED
  on(4);         // blink pin 4 on and off
  pause(500);
  off(4);
  pause(500);
```

The whole process looks like this:

1) Start from the very beginning and insert the LED blink code to establish that uploading is working, and that the blink code works.
2) Move the blink code a few lines down and upload the code again.
3) If the blink code didn't trigger, look before that point for clues on why it didn't get there.
4) If the blink code did trigger, move the blink code a little further down in the program and upload again.
5) Repeat steps 2-4 until you've found all your bugs!



Finally, debugging a program is not something you have to do alone. **Try explaining the problem you are having out loud to someone else.** Often you will catch the issues even before you finish explaining. If not, the other person may be able to offer suggestions or ask helpful questions. This technique is effective even if the other person hasn't coded before! In fact, some progammers use a technique called "rubber duck debugging" where they first try explaining their problems to a rubber duck.



Instead of simply saying "my code is not working" or "my code is broken!", break your story down into specific intentions and observations: "this is what I expected my code to do, but it's doing this instead." This helps clarify what you're trying to debug, and will also make it easier for someone else to help you.

CODE (CONT'D)

Debugging is an important programming skill! Don't worry if it takes you a while to solve a problem, you are building useful skills in the process, while growing to understand coding better!

...AND BEYOND

The debugging suggestions we shared in this chapter are only some of the most common tips meant to help you get started with the debugging process. For more debugging resources, check out our website at **chibitronics.com/lovetocode.**

Don't worry if it's frustrating at first. Debugging is a skill that takes some patience but you get better at it as you debug more projects. It can also be fun to create (and debug) with a friend. Fresh eyes can catch things that we don't see ourselves, so reach out and ask others for help too!