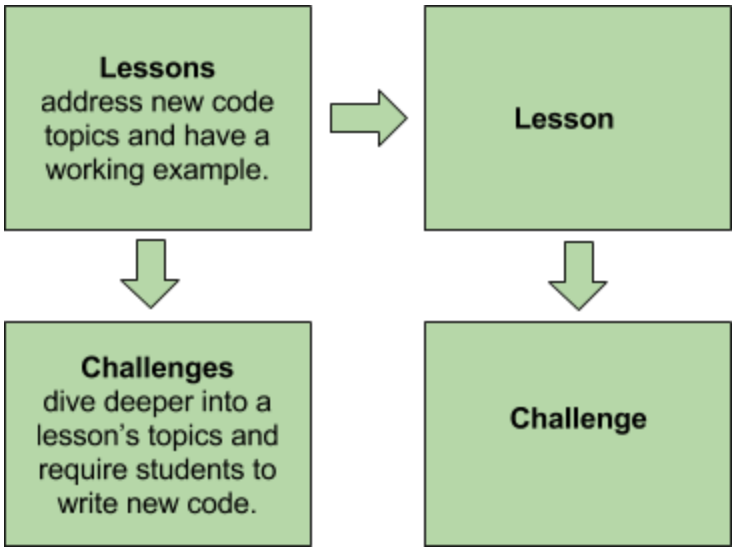
Lesson Plans & Quizzes with Base Kit
14 (1-hour) lessons with two bug hunt activities

Find the lessons at http://www.letsstartcoding.com/learn

Introduction

This curriculum includes 14 coding lessons that use the *Let's Start Coding Base Kit*. Each lesson is expected to take approximately 1 hour to complete.

Each lesson includes a **code challenge**. It's an opportunity for students to stretch their skills and try to modify the lesson's code for a new result. The challenges can be used for students who are moving more quickly than the rest of the class, as a second lesson on a particular coding topic, or as homework.



Teachers very new to teaching programming may wish to use only the lessons, removing the chance of finding a problem that they cannot solve. Challenges, though, are designed to be achievable given the skills taught by the lesson. If you have the time, we recommend completing a lesson, then its challenge, then the next lesson.

Quizzes

Each lesson is followed by a short quiz that can be answered with knowledge of the previous lesson. The quiz questions are not easy, but it's important to have an understanding of their answers since the concepts aren't usually covered again in later lessons.

Many of the answers to the questions can be completed with an experiment. We highly encourage you to allow students to experiment with their code to find the answers. An important skill in coding is the ability to design and run an experiment. The questions that can be answered with a hands-on experiment are marked with our smiley chip logo .

Time Management

Each lesson follows the same time structure. You can reduce setup time by laying out the appropriate component cards for your students and by opening their internet browsers to the appropriate project page in a Chrome browser.

Notes	**Activity**	**Approximate Time Allotment**
These activities can be prepped ahead of time to increase coding time.	Get computers started and upload a test program on Maker Board	7 min
	Navigate to the web page for the appropriate lesson	5 min
	Read the example program and develop hypotheses about how it works	3 min
	Upload the lesson code to Maker Board and observe it	5 min
	Read the code walkthrough and new concepts explanations	10 min
	Tinker and experiment with the code by editing it	15 min
	Record learning in the programmer's notebook	10 min
Optional	Share code creations with classmates	*5 min*
	Try the code challenge for the lesson	*30 min*
	Record hypotheses, errors, and solutions to challenge	*10 min*
	Share challenge solutions with classmates	*15 min*
	Clean up and store kits	5 min

In total, you should plan for 60 minutes of classroom time per lesson. The optional steps will extend your students' learning and can add up to an hour of time to each lesson.

Contents

Contents (continued)

Pre Activity 1: What is Code and How Do I Read It?

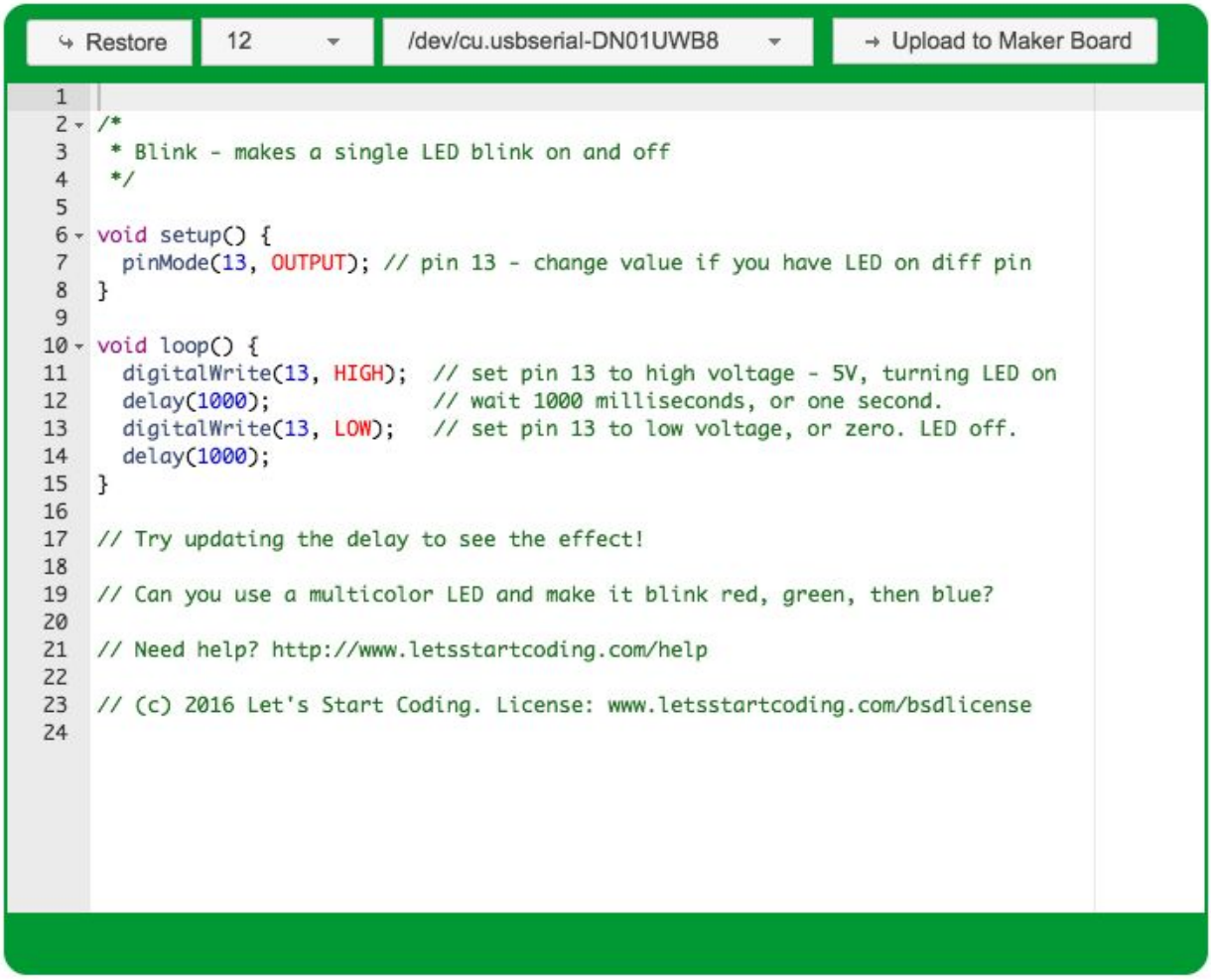
(Teacher Script or Student Reading)

Code, simply, is instructions that humans write for computers. Computers are fast and tireless, but they are not ‘smart’. Writing code allows humans to use computers as useful tools. Keep in mind as you look at code that some other *person* wrote it- it isn’t magic!

Like a written language, coding languages have rules. Some things must be capitalized. Punctuation is required in the right places. You will learn these rules by looking at code and by reading documents online that explain code.

When you first look at the code, you’ll see lots of words, shapes, numbers, and colors that don’t make sense. Believe it or not, those are designed to help you! Even though it is confusing at first, you should carefully read through the code that you see in the lessons. Eventually you will notice patterns that help you write better code.

Let’s look at a program and dissect the parts.



Within this simple code, you can see 5 different text colors and 6 different punctuation (syntax) marks. Each of those means something and, when you know what they mean, make the code easier to read.

Comments

Text in green is called a comment. It is *not* code for the computer- it's a note written by the previous programmer to be viewed by you! Even if you understand nothing else in the code, read the comments. They should explain the intent of the code.

Comments have to be 'marked' so that the computer doesn't try to use them as code. To mark the beginning of a multi-line comment, you type /*. The words following that mark are excluded from the code until you type */. To mark a single line comment, type //. The rest of that line will be excluded from the code. In the code above, the first comment tells you the title of the program and what it's supposed to do- pretty handy!

Keywords

The other non-black colors in the code indicate keywords. Some words come up again and again, so we automatically highlight those terms to make it easier to find mistakes. For example, if you type HIHG, the text will *not* turn red. Once you've learned to skim code, the colors will pop out at you and you'll be able to find errors.

Syntax

The punctuation marks you see in the code are a part of the *syntax* of code. You use this when you write, too. Sentences end with periods, question marks, or exclamation marks. Paragraphs are sentences that belong in a group. Code is the same.

When you see an open parenthesis, it will be matched with a closing parenthesis. When you see an opening curly brace, it will be matched with a closing curly brace. The code between two parentheses or between two curly braces has been grouped together by the coder. In the code above, you see that there is an opening curly brace on line 10. There is a closing curly brace on line 15. Everything between those two is grouped together in the void loop().

There can be groups within groups within groups in code. That's why you'll see indentation in the code. The indentation doesn't affect how the computer runs the code. It's a way for programmers to organize their code to make it easier to read. Remember that we said everything between lines 10 and 15 is grouped together? Look at how they are indented. Because lines 11,12,13, and 14 are all indented the same amount, you can make an educated guess that they are grouped together, even if you aren't sure what they do.

All programs are made up of small building blocks that are pieced together. It's important that you start reading the programs while they are small and have only a few pieces. As programs get larger and more complex, you'll be able to see the small pieces that make them up.

If you have time after the Pre Activity, you can set up your computers to program Maker Board. Ideally, you will have set up at least one student computer using student login credentials and permissions. This will ensure that your system administrator has not blocked any of the necessary parts of the installation process.

Students can navigate to **http://www.letsstartcoding.com/start** to set up their web browsers to code on www.letsstartcoding.com. Ensure you're using a Google Chrome web browser.

Pre Activity 2: What Are My Components and How Do I Use Them?

Components are the hand-held parts of the kit. The things inside the box that interact with the code to make something *real* happen. Unlike the code, there is no “undo” button for real life objects, so it’s important to understand how to use (and how *not* to use) the components in your kit.

Identifying Your Components

Each component in your kit comes with a corresponding card. The card identifies the component, shows you how to plug it in, and warns you about activities that may damage the component. You should read through each of these cards carefully before using your kit.

Look out for the + symbol!

There is a + symbol on the mini carrier board in 6 different locations. Plugging a component into a + port is the only way you will damage a component with electricity. Some components require the +, but many do not. Locate each of the + symbols on the carrier board.

It is a near-certainty that someone will plug an LED light into the + socket early in the class. The + is very close to the 13 port, which is used in the programs. No project in the kit requires *every* LED light, so any kit is still functional while missing an LED. Here are some strategies to avoid breaking any components:

- Do not connect Maker Board to power when you are plugging in components. Maker Board slides out of Carrier Board easily, so you do not need to leave it plugged in. Remove Carrier Board, plug in your components (hold the Carrier Board close so you can see it) and double check that no components are plugged into the +. Only then plug in Maker Board and power the gadget.

- If a student does plug a component into the + port and breaks it, remove power from the board. Now show the student where their component is and why it broke. Many students’ first inclination is to remove the component from the port, but then they do not believe they plugged the component in to a + port. With the power removed from the board, the component is no longer under pressure and all of the students can learn from the mistake.

- If it is very important to you not to break any components, consider placing a piece of tape over the + ports of the Carrier Board to consistently remind students that it is not the port they intend to use.

Errors in plugging in components will decrease dramatically once students have a little experience with the kits. However, it is difficult to overemphasize that the components can break during the first few lessons.

Using the Cards in the Kit

While they aren't electronics, the cards in the kit are your easy-to-use resources when it comes to coding and using the electronics. Kits should include a card for each component. They also include introduction cards and cards about common code concepts. When you are working on a program and you are struggling, look for a card to help! Take some time to go through the card deck and recognize the different colors and types.

Lesson One: Blink

This lesson introduces students to a number of new concepts. First, code on the screen can make real-world things happen. The speed of the blinking light is directly affected by the code students write. Second, students learn how a program 'loops' or repeats- following the instructions that they give it an infinite number of times.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board. We recommend that you set up each computer using the "Quick Start with Codebender" setup guide on our website. Setting up each computer with this method should take under 10 minutes. Setup can be completed by students or by educators.

It is a good practice for students to first sit down at their computer and upload "Blink" to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected. Because this lesson's program *is* Blink, this step can be skipped.

Find the Project Page and Plug in the Components- 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Blink lesson page from /learn or navigate there directly at www.letsstartcoding.com/learn-blink . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students hook up a blue LED to pin 13 on the Carrier Board. They will need to look at the component card for the LED and try to figure out how to plug it into the Carrier Board.

Read the Example Program and Identify its Parts- 3 min

Students should be able to identify what the example program on the page does by reading the comments. They will also start to identify, using only the code, where the components are plugged in on Carrier Board.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

When students are uploading code for the very first time, there will likely be some kind of failure. Ask students who are having trouble to get help from a student who has a successful upload. The most common errors are selecting the wrong port or an incomplete installation of the Codebender plugin. This error will result in the code window graying-out the 'Run on Arduino' button and disabling the drop-down windows.

If this happens, ensure that you're using an up-to-date Chrome browser and that plugins are enabled. If you are using the Chrome browser and plugins are enabled,, restart the install process at www.letsstartcoding.com/start.

Probing Questions for Students:

- *Try to count how fast the light is blinking. How long is it on? How long is it off? Can you relate that speed to anything you see in the code?*

Read Code Walkthrough and New Concepts - 10 min

After students have uploaded the code and have their light blinking on Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code.

If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing Questions for Students:

- *Why do you think the code mentions the number 13? What happens if you change that?*
- *What happens if you turn a line of code into a comment by placing // in front of the line?*
- *How are the two 'delay(1000)' lines the same? How are they different?*

After a few minutes of play, you should have some students with lights blinking extremely slowly (or not at all) and some students with lights blinking so fast it appears that they are permanently on. Students will experiment with the threshold at which the light no longer appears to be blinking but dimly lit all of the time (this is related to the science of persistence-of-vision!).

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 1

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson One Quiz : Blink

How many times will the LED blink when running the example code?

A. 10 times
B. 100 times
C. 15 times
D. Forever

Answer: D. The void loop will continue to run until power is removed from Maker Board.

If you make the value of both delays smaller, what will happen?

A. The LED will blink faster
B. The LED will blink slower
C. Nothing changes

Answer: A. A smaller delay is a smaller 'wait time', so the loop will run faster.

In the example *digitalWrite(13,HIGH);* what are the *arguments*?

A. digital and Write
B. HIGH
C. 13 and HIGH
D. 13

Answer: C. The values between the parentheses are the arguments of a function.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

In lesson 1, the code challenge requires students to plug in a new component (multi-color LED) and use the digitalWrite() command with new pins. This is a good time to practice using components carefully and referencing the cards in your kits.

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms

of coding- “I added a delay(1000); to the code”- or in terms of thinking - “I remembered that the loop will run very quickly without code to slow it down”.

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Two: Play 3 Tones to Speaker

This lesson introduces students to a new component and explains further the underlying structure of these programs: some things happen only once and some things happen over and over. These are functions delineated by curly braces.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board. We recommend that you set up each computer using the "Quick Start with Codebender" setup guide on our website. Setting up each computer with this method should take under 10 minutes. Setup can be completed by students or by educators.

It is a good practice for students to first sit down at their computer and upload "Blink" to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Play 3 Tones to Speaker lesson page from /learn or navigate there directly at www.letsstartcoding.com/learn-blink . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the speaker and try to figure out how to plug it into the Carrier Board.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. They will also start to identify, using only the code, where the components are plugged in on Carrier Board.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *What unit of measure do you think the program is using for sound?*

Read Code Walkthrough and New Concepts - 10 min

After students have uploaded the code and have their speaker 'chirping', ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code.

If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing Questions for Students:

- *Why do you think the code only runs one time?*
- *What happens if you 'comment out' the line containing 'noTone'?*
- *Can anyone find the musical note that corresponds to each tone using the internet?*
- *What portions of the code do you need to move to make the tones repeat? What parts need to stay put?*

Students will typically change the value of the tone commands to extremely high or low values. They will often ask if the speaker can get louder- it cannot. If students are familiar with older video games like Mario, you can tell them that this technology is similar to what those creators worked with when they created those iconic themes.

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 2

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Two Quiz : Play 3 Tones to Speaker

What symbols (called syntax) begin and end the *void setup* function?

A. (and)
B. { and }
C. ;
D. //comments

Answer: B. Curly braces define the beginning and end of the contents of a function.

What syntax begins and ends the *void loop* function?

A. (and)
B. { and }
C. ;
D. //comments

Answer: B. Curly braces define the beginning and end of the contents of a function (for *all* of the functions we'll look at!).

If you type:

```
tone(A5,500);
noTone(A5);
```

how long will you hear the tone of 500 hertz?

A. 1 second
B. 10 seconds
C. You won't hear it at all
D. You'll hear it continuously

Answer: C.Without a delay, the code runs too fast for you to see any single loop or line of code.

Optional: Bug Hunt One!

We created bug hunts to encourage students to critically read code and to take away the stigma of 'being wrong' when you have a bug in your code. A study at Kent University studied over 37 million compilations (attempts to run) of code and found that over 17 million of them resulted in an error. Errors aren't something to be afraid of- they're puzzles to be solved.

This bug hunt uses the same project that students just played with in the lesson, so they aren't tricked by a new concept that may *look* like a bug but not actually be one. It also addresses some of the most common errors when beginning to code.

The inclination when students have a coding error is to fix it- change *something* and try again. The bug hunt encourages students to slow down, think about the error, then find the solution.

Encourage students *not* to try to run the code, but just read through it, looking for things that look strange to their eye. When they have found 3 errors, have them write the errors in their Programmer's Notebook. By asking them to write the error, you're asking them to articulate the problem, not just notice a difference.

Next, have the students press "Run on Arduino". The computer will point out the first error it finds. Assuming the students found the error correctly, have them look at the warning from the computer versus the error they found.

The computer gives you clues that are usually helpful in finding an error, but you have to learn to read and follow the clues. Tying a 'clue' to the error in the code makes it easier to follow that clue next time you see it.

Now have students correct the first error they found. Press "Run on Arduino" again. If they fixed the error correctly, a new error should come up. Repeat this process of observing and fixing errors until the program runs. If students missed an error, they should note it in their programmer's notebook.

A note on bug hunts: It's very easy to insert bugs into code. If you have a projector available, it can be a fun and helpful activity to invite students to introduce a bug they don't think their classmates can find.

Here's how to do it: Open the working code on the projector- everyone should be familiar with it. Invite one student up to the computer controlling the projector and then 'freeze' the projector's image so that no other student can see the code change. Have the student make her change to the code, then put the cursor back at the top of the code so that it isn't obvious which area she modified. Now unfreeze the projector- the rest of the class will see the now-buggy code.

Have the class try to identify the bug. Once it's found, have everyone watch as you press "Run on Arduino" and see the compiler error. Now you can tie the error clue to the error.

The goal for the bug-inserter is to get through as many guesses as she can with her bug unseen. The goal for the class is to find it as quickly as possible. Bug hunts are a great way to fill 5-10 minutes and get the entire class to work together.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

In lesson two, the code challenge requires students to copy and paste code from the void setup() to the void loop(). Remember that these functions start with a curly brace and end with a curly brace. Students will often copy and paste too much of the code and get a strange error.

Emphasize for these students that they need to pay very close attention to the code, because the computer cannot interpret what they ‘mean to do’.

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding- “I added a delay(1000); to the code”- or in terms of thinking - “I remembered that the loop will run very quickly without code to slow it down”.

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

This particular lesson will likely end with lots of different songs, themes, and jingles being created. Students love to create their favorite jingles!

Lesson Three: Hold Button to Turn LED On

In this project, students get a very simple exposure to a very important concept- the idea that code can behave differently depending on inputs. That is, code can react and interact with other code, the environment, or, in this case, the programmer himself!

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload “Blink” to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Hold Button To Turn LED On lesson page from /learn or navigate there directly at www.letsstartcoding.com/learn-hold-button-to-turn-led-on/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the LED and button and try to figure out how to plug each into the Carrier Board. Keep in mind that the button is not polarized- you can plug either leg into ground and the other leg into the signal port.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. They will also start to identify, using only the code, where the components are plugged in on Carrier Board.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *How fast can the light turn on and off?*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. Remind them that learning these topics now will make their programs much more interesting in the future.

If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing Questions for Students:

- *What part of the code is a condition? What part is the dependent outcome?*
- *Can you frame this program in terms of dependent and independent variables?*
- *What do you think all of the curly braces are for? Can you identify which pairs up to the other?*
- *Can you reverse the program so that the light is on unless the button is held?*

- *Can you move the components around so that the button is in a new port and the LED light is in a new port?*
- *Can you add a delay so that the light pauses in the 'on' state after a button press?*

This is a good chance for students to play with the components and make changes in their hardware. They can switch which color of LED light they use, move the button to a new location, or both. They can also learn to reverse the action of the program or 'comment out' the 'else' statement so that the light stays on once the button is pressed.

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer’s Notebook: Lesson 3

Read one of the new concepts from the web page containing this lesson’s code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Three Quiz : Hold Button to Turn LED On

Which symbols (called comparators) check to see if two things are exactly equal to each other? How did you check to see if the digitalRead was equal to LOW?

A. ==
B. =
C. +=
D. ><

Answer: A. The == checks for equality. The single = *assigns* equality.

How many arguments does the digitalRead function require? 

A. 1
B. 2
C. 0

Answer: A. The only argument that function needs is the pin to take a reading from.

True or False: You must have an *if* statement in order to use an *else* statement. 

A. True
B. False

Answer: A. An *else* statement without an *if* statement will cause an error. Logically, the *else* statement has no conditions to run.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

The code challenge for lesson three requires students to add new components- an extra LED light - to the Carrier Board. Then they will have to comb through the code and find out how to add the code for the second LED.

Many students will miss the pinMode() function for their new LED in the void setup() function. You will see that the program works, but the lights are very dim- that is a clue that they’ve missed the pinMode() function.

Another typical error is that students disregard the curly braces and put their code somewhere outside of the curly braces.

To complete this challenge, students need to find each line that references the first LED, go to the end of that line (past the comment) and press enter to create a new line. They can then copy the same function that used the LED, replacing the original number with their new LED number.

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a

hypothesis each time they upload a program. Even if they're experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press 'upload' many times until they're lost and frustrated.

Similarly, recording errors in the Programmer's Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding- "I inserted new digitalWrite() functions for my new LED"- or in terms of thinking - "I copied the relevant code and made sure it stayed inside the curly braces with the other similar statements".

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

This particular lesson is a good opportunity for students to share the problems they faced when completing the challenge- the outcomes may look similar, but it's likely that students took many different routes to get there.

Lesson Four: LED Flashlight

This lesson will appear very familiar to students, as it takes the same components from the previous lesson and adds some functionality. However, this lesson exposes students to a concept they may have seen in math, but used differently: the variable. For most beginner students, code variables are a new paradigm, so encourage them to read carefully as they go through this lesson.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload “Blink” to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the LED Flashlight lesson page from /learn or navigate there directly at www.letsstartcoding.com/learn-led-flashlight/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the LED and button and try to figure out how to plug it into the Carrier Board. Keep in mind that the button is not polarized- you can plug either leg into ground and the other leg into the signal port.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. They will also start to identify, using only the code, where the components are plugged in on Carrier Board.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *When you first upload the program, what is the value of the variable 'pressCount'?*
- *What is the value of the variable 'pressCount' after you've pressed the button once? Twice?*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing Questions for Students:

- *What happens if you set pressCount equal to 1 when you create it near the top of the program?*
- *What happens if you remove one of the equals signs from an == statement?*
- *Can you rename the variable? What names are acceptable? Can you find any names that aren't?*
- *What happens if you hold the button down, in coding terms? What happens if you change the delay in the program?*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 4

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Four Quiz : LED Flashlight

Because the variable is outside of any function (above void setup() and void loop()), you can refer to it as a _____ variable.

A. Global
B. Local
C. Static

Answer: A. Global variables are declared outside of all functions.

Which of these is the best definition of variable?

A. A code command for electronics.
B. A word that holds a value.
C. A number in the code.

Answer: B. Variables are represented with words or letters, but they have number values.

What goes between the parentheses of an *if* statement?

A. The condition that is being checked by the *if* statement.
B. The code that will run when the *if* statement is true.

Answer: A. The condition is checked in the parentheses, the commands are between the curly braces.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

Like the previous challenge, students can use the existing code to see how to add on to it. One of the underlying ideas with the challenge is that students will start to understand just how fast the Maker Board can move.

By placing a statement or line of code just below another one, the code will run near-simultaneously. However, the computer reads the program from top to bottom, so there is, in fact, an order in which the statements run. By completing this challenge and others like it, students will experience how fast the code runs and understand why it’s necessary to insert delays into programs.

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own

improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding- “I inserted new digitalWrite() functions for my new LED”- or in terms of thinking - “I copied the relevant code and made sure it stayed inside the curly braces with the other similar statements”.

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Five: Cycle LED Colors with a Button

This lesson builds further on the LED lights and button programs. By learning about the else-if statement, students can expand the number of 'decisions' that their program can make. By the end of this lesson, the students should understand that their programs can have limitless possibilities.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload "Blink" to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Cycle LED Colors with a Button lesson page from /learn or navigate there directly at http://www.letsstartcoding.com/learn-cycle-led-colors-with-a-button/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the multicolor LED and button and try to figure out how to plug them into the Carrier Board. Keep in mind that the button is not polarized- you can plug either leg into ground and the other leg into the signal port.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. They will also start to identify, using only the code, where the components are plugged in on Carrier Board.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *Why do you have to write digitalWrite(10,LOW); at the top of the loop? Hint- it has to do with what happens the second time the loop runs.*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing Questions for Students:

- *Can you make the LED lights cycle without pressing the button?*
- *What happens if you delete the 'pressCount = 0;' statement near the bottom of the code?*
- *How does the delay affect the program? What happens if you change it?*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer’s Notebook: Lesson 5

Read one of the new concepts from the web page containing this lesson’s code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Five Quiz : Cycle LED Colors with a Button

What is the *maximum* number of *else if* statements you can string together?

A. Only one *else if* statement per *if* statement.
B. Unlimited *else if* statements per *if* statement- string together as many as you want.
C. Up to ten *else if* statements per *if* statement.

Answer: B. There is no limit to the number of conditions you can test.

When does an *else if* statement get run?

A. Every time the loop runs.
B. Only when the *if* statement it is paired with is false.
C. Only when the *if* statement it is paired with is true.

Answer: B. *else if* is an alternative to the *if* statement above it.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

This challenge is a good time to encourage students to type their own modified code rather than just copy and paste. Using the example code that they can see, they should be able to type their new else-if statements and get the syntax correct. This will encourage them to practice their keystrokes-some of which are not common in other typing.

Good questions to ask during the challenge:

- *Remember that each of the shorter legs of the multicolor LED represents a different color. How would you make purple? Hint- multicolor LEDs can have multiple colors on at once.*
- *What is the trick to ensuring that the pressCount restarts after the final ‘else if’ statement?*

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer's Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding- "I inserted another else-if statement for the new condition"- or in terms of thinking - "I provided the code another option or path to take".

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Six: Four Note Piano

For this project, students will apply some of the coding concepts they've learned this far to a new set of components- four buttons and a speaker. This is an important idea in coding: the concepts are universal. Turning 'something' on could refer to a light, a fan, a buzzer, a laser, a motor, or any number of other things. The underlying concept won't change much- just the details.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload "Blink" to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Four Note Piano lesson page from /learn or navigate there directly at http://www.letsstartcoding.com/learn-four-note-piano/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the speaker and button and try to figure out how to plug it into the Carrier Board.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. They will also start to identify, using only the code, where the components are plugged in on Carrier Board.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *How is this similar to your previous programs? How is it different?*
- *Do you recognize every statement and function in the program? Which ones don't you recognize?*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing Questions for Students:

- *What happens if you change the numbers in the digitalRead() statement? Hint- it changes which key plays which note.*
- *What happens if you comment out the else statement? Don't forget to comment out the closing curly brace for that statement!*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 6

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Six Quiz : Four Note Piano

How many arguments does the digitalRead() function require?

A. 1
B. 2
C. 3

Answer: A. digitalRead() only needs to know which pin to get a value from.

In Four Note Piano, what happens if you pressed button 4 and button 12 at the same time?

A. The tone for button 4 would play.
B. The tone for button 12 would play.
C. Both tones would play.
D. Neither tone would play.

Answer: A. The tone for button 4 is checked by the first *if* statement. If that statement is true, none of the *else if* statements are checked.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

This challenge is a good time to encourage students to type their own modified code rather than just copy and paste. Using the example code that they can see, they should be able to type their new else-if statements and get the syntax correct. This will encourage them to practice their keystrokes-some of which are not common in other typing.

Good questions to ask during the challenge:

- *Where do you have to create a variable? Refer back to the previous programs that use variables.*
- *How can you replace a number with a variable?*

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer's Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding- "I created a variable and inserted into the tone function"- or in terms of thinking - "I replaced a number with the variable".

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Seven: Four Note Piano with Synced LED

As your students have seen in previous lessons, there is more to code than just memorizing statements and perfecting typing. The structure and order of a program also has a big effect on how well the program works. As the complexity of the code grows, style (like indentation) becomes more important to keep track of the logic in a program.

This lesson starts to expose students to those ideas as you add in a familiar component- the LED light- to a past project- the piano.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload “Blink” to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Four Note Piano with Synchronized LED lesson page from /learn or navigate there directly at http://www.letsstartcoding.com/learn-four-note-piano-with-synchronized-led/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the speaker, LED, and button and try to figure out how to plug each into the Carrier Board.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. They will also start to identify, using only the code, where the components are plugged in on Carrier Board.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *Compare this program to the last program- what is different?*
- *Can you tell why the indentations are in the code?*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing questions for students:

- *Can anyone look up which notes correspond to which hertz using the internet?*
- *What happens if you remove the noTone() function?*
- *Which tones sound like a normal piano?*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect

the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Lesson Seven Quiz : Four Note Piano with Synced LED

How many code commands can be run inside an *if* statement?

A. Ten commands maximum.
B. Two commands maximum.
C. Unlimited number of commands.
D. One command maximum.

Answer: C. You can string together as many commands as you want.

When you type the code:

tone(A5, 250);
digitalWrite(2, HIGH);

which command runs first?

A. They run at exactly the same time.
B. The tone() command runs first because it is above the digitalWrite() command.
C. The digitalWrite() command runs first because it has a lower pin number.

Answer: B. Technically code runs from top to bottom, although these commands will seem to run at the same time because code is so fast.

When does an *else* statement run?

A. When all the *if* and *else if* statements in the same chain are false.
B. Every time the loop passes over it.
C. Never.

Answer: A. When all of the other alternatives are false, the *else* statement runs.

Note: We say 'in the same chain' because it's possible to have multiple *if-else if-else* chains in a program. If three buttons controlled the color of an LED and three other buttons controlled the tone on a speaker, those would be separate chains of *if-else if-else* statements.

Programmer's Notebook: Lesson 7

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Optional: Bug Hunt Two!

We created bug hunts to encourage students to critically read code and to take away the stigma of 'being wrong' when you have a bug in your code. A study at Kent University studied over 37 million compilations (attempts to run) of code and found that over 17 million of them resulted in an error. Errors aren't something to be afraid of- they're puzzles to be solved.

This bug hunt uses the same project that students just played with in the lesson so they aren't tricked by a new concept that may *look* like a bug but not actually be one. It also addresses some of the most common errors when beginning to code.

The inclination when students have a coding error is to fix it- change *something* and try again. The bug hunt encourages students to slow down, think about the error, then find the solution.

Encourage students *not* to try to run the code, but just read through it, looking for things that look strange to their eye. When they have found 3 errors, have them write the errors in their Programmer's Notebook. By asking them to write the error, you're asking them to articulate the problem, not just notice a difference.

Next, have the students press "Run on Arduino". The computer will point out the first error it finds. Assuming the students found the error correctly, have them look at the warning from the computer versus the error they found.

The computer gives you clues that are usually helpful in finding an error, but you have to learn to read and follow the clues. Tying a 'clue' to the error in the code makes it easier to follow that clue next time you see it.

Now have students correct the first error they found. Press "Run on Arduino" again. If they fixed the error correctly, a new error should come up. Repeat this process of observing and fixing errors until the program runs. If students missed an error, they should note it in their programmer's notebook.

A note on bug hunts: It's very easy to insert bugs into code. If you have a projector available, it can be a fun and helpful activity to invite students to introduce a bug they don't think their classmates can find.

Here's how to do it: Open the working code on the projector- everyone should be familiar with it. Invite one student up to the computer controlling the projector and then 'freeze' the projector's image so that no other student can see the code change. Have the student make her change to the code, then put the cursor back at the top of the code so that it isn't obvious which area she modified. Now unfreeze the projector- the rest of the class will see the now-buggy code.

Have the class try to identify the bug. Once it's found, have everyone watch as you press "Run on Arduino" and see the compiler error. Now you can tie the error clue to the error.

The goal for the bug-inserter is to get through as many guesses as she can with her bug unseen. The goal for the class is to find it as quickly as possible. Bug hunts are a great way to fill 5-10 minutes and get the entire class to work together.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

This challenge is a good time to encourage students to type their own modified code rather than just copy and paste. Using the example code that they can see, they should be able to type their new else-if statements and get the syntax correct. This will encourage them to practice their keystrokes-some of which are not common in other typing.

Good questions to ask during the challenge:

- *How will you ensure that all of the lights turn off when you release a key?*
- *Can you use a multicolor LED? Can you blend two colors for one note?*
- *Can you change the note a variable and complete the LED challenge?*

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own

improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding- “I created a variable and inserted into the tone function”- or in terms of thinking - “I replaced a number with the variable”.

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Eight: Button-Activated Siren

Now that students have an understanding of variables and what they can do, it's time to stretch their understanding further. In this lesson, a pitch variable will rise and fall as the students press or release a button. The significant lesson here is that the students don't always know what the value of the variable is. They're really taking advantage of the computer's speed and number-tracking abilities to make something fun.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload "Blink" to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Button-Activated Siren lesson page from /learn or navigate there directly at http://www.letsstartcoding.com/learn-button-activated-siren/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the speaker and button and try to figure out how to plug it into the Carrier Board.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. They will also start to identify, using only the code, where the components are plugged in on Carrier Board.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *What would be another way to write pitch = pitch +1 ? Hint: It's pitch ++;*
- *What is the purpose of the final 'if' statement in the code?*
- *How often will a tone be played?*
- *What is the value of tone the first time through the loop? The second time? The third time?*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing questions for students:

- *Can you make the tone rise more quickly? Fall more quickly?*
- *Can you reverse the effect of the button, making the tone fall when the button is pressed?*
- *Can you make the tone stay flat when the button is released?*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 8

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Eight Quiz : Button-Activated Siren

What is a different way to code *variable = variable + 1;* ?

A. variable ++1;
B. variable +1;
C. variable ++;

Answer: C. ++ is an incrementor that you first saw in 'LED Flashlight'.

In the button-activated siren example, what is the lowest value that the *pitch* variable can have?

A. 0
B. 40
C. 41
D. 5

Answer: B. The *if pitch <=40* code statement will set *pitch* equal to 40 every time the condition is true.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

Good questions to ask during the challenge:

- *What existing code ‘patterns’ or concepts will you use to complete the challenge?*
- *Why is it useful to put a limit on the pitch?*
- *What devices in your own life can you relate to this program?*

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms

of coding- “I created a variable and inserted into the tone function”- or in terms of thinking - “I replaced a number with the variable”.

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Nine: Button-Activated LED Fade

This lesson parallels the last, demonstrating the the crossover that concepts can have. By repeating fundamentals in a new context, students not only strengthen their skills but start to imagine more scenarios where they could apply the concepts.

In the last lesson, the sound was what 'faded' up and down. Here, students fade a light up and down, using the constrain function to keep the LED brightness within a certain bounds.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload "Blink" to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Button-Activated LED Fade lesson page from /learn or navigate there directly at http://www.letsstartcoding.com/learn-button-led-fade/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the LED light and button and try to figure out how to plug each into the Carrier Board.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. They will also start to identify, using only the code, where the components are plugged in on Carrier Board.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *What would be another way to write brightness = brightness +1 ? Hint: It's brightness ++;*
- *How does this program limit the fade differently than the last program limited the pitch?*
- *By using your code concept cards, can you identify which pins work with analogWrite()?*
- *What happens if you hold the button down? Can you tell just by looking at the code?*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing questions for students:

- *What happens if you change the value in the '0' position of the constrain function?*
- *How can you make the light get brighter faster?*
- *How can you make the light make bigger jumps from light to dim?*
- *What happens if you replace the analogWrite() commands with digitalWrite()?*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 9

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Nine Quiz : Button-Activated LED Fade

What is the range of the *analogWrite()* function?

A. 0 to 100
B. 0 or 1
C. 0 to 255
D. 0 to 1000

Answer: C. Writing a value above 255 will 'roll over' to 0 again. analogWrite(257); is equal to analogWrite(1);

What are the arguments needed by the *constrain* function?

A. Pin number and HIGH or LOW
B. Variable, minimum value, maximum value
C. Minimum value, maximum value
D. Pin number and mode

Answer: B. constrain() must know what variable it is limiting and what the limits are for that variable.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

Good questions to ask during the challenge:

- *Why do you need to create the variable outside the void setup() and void loop()? What if you don’t?*
- *Can you use the brightness variable for the tone, too?*
- *Can you create code so that a separate button causes tone to rise?*

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding- “I created a new integer and inserted it into the if statements”- or in terms of thinking - “I discovered that I just had to copy/paste the existing code with very few changes.”

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Ten: Automatic Siren

In this lesson, students discover a new way to do something that seems familiar. Taking the siren from button-activated to automatic requires a new tool in their toolbox. By this point, students may have already recognized the need for this new concept - the for loop - without knowing it. When they see it in action, they'll likely be able to apply it to their earlier projects.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload "Blink" to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Automatic Siren lesson page from /learn or navigate there directly at http://www.letsstartcoding.com/learn-automatic-siren/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the speaker and plug it into the Carrier Board.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. By this point, they can likely identify the position of the components by reading the void setup() of the program.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *Can you calculate the time it should take for the tone to rise to its maximum?*
- *What is an alternative way to write pitch++ or pitch - -?*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing questions for students:

- *Can you trigger the siren to turn on and run with a button? Can you make it turn off?*
- *Could you accomplish this program with if statements? What would that look like?*
- *Can you change the starting point or ending point of the pitch in each for loop? What if they don't 'match up'?*
- *How can you make the siren increase its pitch by more than 1 hertz each time the for loop runs?*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 10

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Ten Quiz : Automatic Siren

Which of the following is *not* a part of creating a *for* loop? 

A. Variable with starting value.
B. Condition to test each time the *for* loop runs.
C. Code to modify the variable each time the *for* loop runs.
D. Empty parentheses ()

Answer: D. There shouldn't be empty parentheses in a *for* loop.

How many times will a *for* loop run? 

A. An infinite number of times.
B. Until it satisfies the condition you give it as one of the arguments.
C. 100 times.
D. One time.

Answer: B. The second argument of the *for* loop defines when the *for* loop breaks.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

Good questions to ask during the challenge:

- *Can you insert an if statement into the for loop so that if the value of pitch is divisible by 10, the LED turns on, otherwise it’s off?*
- *What are ways to make a better looking siren with the LEDs?*
- *Can you use analogWrite() commands to fade the LED up with the tone?*

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding- “I created a new integer and inserted it into the if statements”- or in terms of thinking - “I discovered that I just had to copy/paste the existing code with very few changes.”

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Eleven: Automatic LED Fade

This lesson uses similar concepts to lesson 10, applied to a an LED light. The goal is to reinforce the different ways to use the code fundamentals and to expose students to concrete examples of the fundamentals in action.

By the time students complete this lesson, they should have a grasp on how code can have 'loops within loops' and have an idea of how a variable can change value without their knowing exactly what that value is.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload "Blink" to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Automatic LED Fade lesson page from /learn or navigate there directly at http://www.letsstartcoding.com/learn-automatic-led-fade/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the LED and plug it into the Carrier Board.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. By this point, they can likely identify the position of the components by reading the void setup() of the program.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *How many times do you think the brightness variable will change within one loop of the void loop? Hint: 255+255.*
- *Using your code concept cards, explain why the for loop stops at 255.*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing questions for students:

- *Can you modify the code to brighten the LED bulb to half-brightness?*
- *Can you insert two buttons: one that triggers the LED to fade all the way up with a press and one that triggers the LED to fade all the way down with a press?*
- *Try changing the variable name for one of the for loops to understand that they are not the same variable.*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 11

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Eleven Quiz: Automatic LED Fade

Which code commands can you use inside the curly braces of a *for* loop?

A. Only analogWrite() and delay()
B. Only digitalWrite() and digitalRead()
C. Only tone() and noTone()
D. You can use any code commands inside a *for* loop

Answer: D. *For* loops help control the flow of your code- you can put any valid code statements inside of them.

True or False:

When you create a variable inside a *for* loop, it only exists inside that *for* loop.

A. True
B. False

Answer: A. Variables created inside the parentheses of a *for* loop are called *local* variables, and they can't be used outside of that *for* loop. If you see two *for* loops with the same variable name used for the counter variable, those are actually two separate variables that just share the same name.

Note: This is why you'll often see a variable just called *i* used for *for* loops. It doesn't need a descriptive name because it's only there as a placeholder during the *for* loop.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

Good questions to ask during the challenge:

- *Why can’t you use LEDs in all of the pins? What happens if you try?*
- *Is there a way to use a for loop to set all of your pins to OUTPUT?*
- *Can you create more for loops so that some lights fade up, then down, then other lights fade up, then down?*

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding or train of thought.

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Twelve: Tune LED Colors with Buttons

As the complexity of projects grow, students will start to learn code practices that help simplify and organize their code. In this lesson, they learn about functions- a core coding idea that allows for some bits of code to be repeated simply.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload “Blink” to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Tune LED Colors with Button lesson page from /learn or navigate there directly at http://www.letsstartcoding.com/learn-tune-led-colors-with-buttons/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the LEDs and buttons and plug them into the Carrier Board.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. By this point, they can likely identify the position of the components by reading the void setup() of the program.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *What is happening when you press a button down? Where are the commands to make that happen?*
- *If you were to write a program to do what they see happening in the code, how would you do it?*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing questions for students:

- *How can you add another LED to this program without adding another button?*
- *How can you add an LED and another button to this program?*
- *If you want to increase the brightness more quickly, what do you need to do? What about increasing the color in larger 'leaps'?*
- *Why don't all of the colors change brightness at the same time when any button is pressed?*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 12

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Twelve Quiz : Tune LED Colors with Buttons

When creating a function, where do you define what arguments the function will require?

A. Before the name of the function.
B. Between the curly braces of the function.
C. Between the parentheses of the function.
D. You don't define the arguments of a function you create.

Answer: C. The format of the function you create is: *type name (arguments) {actions}.*

When you call a function from the loop of your code, what code runs after the function has run?

A. The code starts at the top of the loop().
B. The code starts at the top of the setup().
C. The code starts right below the function call in the loop.
D. The code starts right above the function call in the loop.

Answer: C. Think of the function as one line of commands in the loop. After it has run, the code right below the line that called the function will run.

When is a good time to think about creating a function?

A. Every time you write a program.
B. When your program does many different things throughout.
C. They're very rarely useful.
D. When your program repeats the same actions again and again.

Answer: D. Functions remove repetitive code from a loop and make it easier to edit one function to change the entire program.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

Good questions to ask during the challenge:

- *Is the function part of your plan to complete the challenge? Why or why not? Hint: The function isn’t necessary to add the reset button.*
- *What happens if you hold down the reset button and a button to change an LED’s color at the same time? Hint: It depends on the precedent of the else/if statements.*

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding or train of thought.

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Thirteen: LED Dice Roll

Students will create a real-world ‘tool’ or gadget that they can use- a dice! And better yet, it will be fully customizable. On their way to this project, students will find out how to introduce randomness into code and how to create a loop within a loop within a loop within a loop.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload “Blink” to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the LED Dice lesson page from /learn or navigate there directly at http://www.letsstartcoding.com/learn-dice-roll/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the LED and the button and plug it into the Carrier Board.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. By this point, they can likely identify the position of the components by reading the void setup() of the program.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *What is the highest number you've seen so far?*
- *What happens if you hold the button for 10 seconds? 5 seconds? 20 seconds?*
- *What's new in this program? What have you seen before?*
- *What happens if you don't press the button at all? Why do you think that is?*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing questions for students:

- *Can you make a 20-sided dice? 1-sided dice?*
- *How could you 'hack' the dice to roll your desired outcome every time?*
- *Can you make the result show on a different LED than the 'rolling' effect?*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 13

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Thirteen Quiz: LED Dice Roll

What are the possible numbers that random(1,5) could return?

A. 0-5
B. 1-5
C. 0-4
D. 1-4

Answer: D. random() excludes the highest number and includes the lowest number given to it.

When you create code statements inside other code statements, what is that called?

A. Egging
B. Nesting
C. Stacking
D. Enclosing

Answer: B. Think of every statement inside void loop() as 'nested' inside of it. Each time you nest a new statement, indent it one step further than the statement above it. Look at the LED Dice Roll for a good example.

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

Good questions to ask during the challenge:

- *To play the tone while the dice rolls, where will you put the tone variable?*
- *How can you get the tone variable to reset to its starting place after each roll of the dice?*
- *Can you make a ‘success’ tone if the dice rolls a specific number?*

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding or train of thought.

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.

Lesson Fourteen: Sound Samples

The culmination of all of the code concepts learned so far will be a fun set of sound effects. This program is easy to build and fun to play with, but for a student to really be able to tear it apart, modify it, and understand how to recreate it, they need to have a strong understanding of the code.

Get Computers Started and Upload a Test Program on Maker Board - 7 min

Note: This section does *not* include time for the initial setup of the computers to run Maker Board.

It is a good practice for students to first sit down at their computer and upload “Blink” to the Maker Board. This ensures that the computer software is working as expected and the Maker Board is taking programs as expected.

You may also take a moment to remind students of the most common errors and their solutions.

Find the Project Page and Plug in the Components - 5 min

Navigate to letsstartcoding.com/learn to find all of the lessons covered in these lesson plans. Navigate to the Sound Samples lesson page from /learn or navigate there directly at http://www.letsstartcoding.com/learn-sound-samples/ . Each page has a digital card showing which components are required and into which ports they should be plugged.

Have students follow the digital card to connect the components to the Carrier Board. They will need to look at the component card for the speaker and the button and plug them into the Carrier Board.

Read the Example Program and Identify its Parts - 3 min

Students should be able to identify what the example program on the page does by reading the comments. By this point, they can likely identify the position of the components by reading the void setup() of the program.

Ask students to retrieve their code concept cards from the kit. Have them identify which code concepts they see in the program. Students will have used some of the concepts before; others will be new.

Ask some students to share their hypotheses of what will happen when this code runs. This exercise will help them build their coding vocabulary and articulate their thoughts in computer terms.

Upload the Lesson Code to Maker Board and Observe - 5 min

Depending on the interval between lessons, students may repeat some of the same basic mistakes they made when they were coding for the first time. If this happens, it may be worthwhile to review the common errors with students again. If you're using the Programmer's Notebook, you may ask them to write in their upload issues there as a self-reference for the next lesson.

Probing Questions for Students:

- *Can you identify which sound each press of the button makes? How?*
- *Can you see the different parts of the code? What parts of the code seem like they could be deleted without harming the rest of the sound effects?*
- *What's new in this program? Hint: there are multiple new pieces to this code. Boolean AND (||) and the NOT comparator (!=) are two.*
- *How many variables do you count in this program?*

Read Code Walkthrough and New Concepts - 10 min

Once students have uploaded the program onto Maker Board, ask them to read the code explanation on the page that describes what the program does, step-by-step.

Have the students read the rest of the page that introduces the new concepts illustrated by the code. If you are using the Programmer's Notebook, ask students to fill out the vocabulary section *in their own words* describing the topic they just read about.

Tinker and Experiment with the Code by Editing it - 15 min

Probing questions for students:

- *Can you combine any of the sound effects into one?*
- *Can you make the program run all the way through by pressing the button once?*
- *What do you think checkPress() does? Why is it necessary?*

Record Learning in the Programmer's Notebook- 10 min

Have students fill out the questions for this lesson in their Programmer's Notebook. The questions will have varying answers and lengths, depending on the student. You may collect the Programmer's Notebooks and assess your students based on completion of the prompts and/or quality of their answers. Keep in mind that there are very many 'right' answers.

Clean Up and Store Kits - 5 min

If your students have been using the USB cable for connecting Maker Board to their computer, have them carefully wind it back up so that it fits in their code toolbox. If they used any of the cards, be sure they are all packed together and have the elastic band around them. All other components should be placed back in the Code Toolbox. Tell students the box should close without any pressure. If they think they need to apply pressure, something is probably stuck and needs to be moved to fit.

Programmer's Notebook: Lesson 14

Read one of the new concepts from the web page containing this lesson's code. Define the concept in *your own words* and try to explain how it was used in this program.

What surprised you about the program in this lesson?

How did your experimenting go? Were you able to find any fun or cool improvements to the code? What were they?

Lesson Fourteen Quiz : Sound Samples

If you use “&&” between two condition in an *if* statement, what are you doing?

A. Checking that one or the other condition is true to run the statement.
B. Checking that both of the conditions are true to run the statement.
C. Checking that only the second condition is true.

Answer: B. && combines two statements. If both are true, then the result is ‘true’. If either or both of them are false, then the result if ‘false’.

What is the best way to build a complex program or piece of code?

A. Start at the top and go as fast as you can. Check the code when you think it’s finished.
B. Break the problem into pieces, building and testing it one piece at a time.
C. Build it chronologically like you want the program to run. Don’t skip any part or come back later.

Answer: B. When you have a large challenge to solve, it’s probably made up of smaller challenges. Figure out what those are and start to solve them one by one, testing frequently along the way. If you get stuck, it’s OK to go to a different challenge and try to solve it- that may reveal a solution to other challenges!

Optional: Share Code Creations with Classmates - 5 min

Encourage students to ask the person next to them “What did you make your code do?” And “Can you show me how you did that?”. This encourages collaboration and new ways of thinking.

Alternately, you can choose one student volunteer to show their project to the class and describe the process they followed to reach that final project. If space allows or a projector is available, students could gather around and observe the actual lines of code that their classmate wrote.

Optional: Try the Code Challenge for the Lesson - 30 min

In each program, there is a comment at the bottom that encourages the students to try something new with the code. Typically the challenge doesn’t introduce any new code concepts, but suggests a way that the student can copy/paste or modify the existing code to make something new happen.

Good questions to ask during the challenge:

- *Should the off button be in the loop or the function?*
- *What are the different ways you can insert an off button? Why did you pick the one you did?*
- *Could you make an ‘off’ button that resumed the program at the last sound effect when released? What about when you press the other button?*

Optional: Record hypotheses, errors, and solutions to the code challenge - 10 min

Coding is an iterative process and students will often take a step, upload it, observe it, and then take another step toward their goal. That’s a good thing! Encourage students to have a hypothesis each time they upload a program. Even if they’re experimenting, they should have a guess as to what will occur. This process helps students slow down, read their code, and consider it rather than press ‘upload’ many times until they’re lost and frustrated.

Similarly, recording errors in the Programmer’s Notebook is a good practice that will help reduce the repetition of simple mistakes. By recording errors, students are building their own improvement roadmap. Different students make different errors, but writing them down will help reduce them.

Finally, recording solutions can help students think more about their problem-solving skills and identify the flaw in their thinking that led to an error. The solution may be worded in terms of coding or train of thought.

Share Challenge Solutions with Classmates - 15 min

Sharing coding successes and challenges with the rest of the class increases students articulation about coding and their excitement about building something unique. If you have a projector available, showing the code to the rest of the class can reveal different ways of reaching the same solution and spark a Q & A with the student answering questions about their program.