

Educator Guide

Unit 7 – 4 Lessons
60 minutes per lesson
Single Student

Coding with Minecraft: 7- Iteration

EDUCATION.MINECRAFT.NET

THEME OVERVIEW

In this unit, students will explore the coding concept of iteration, or ways to make things repeat. Programmers might repeat actions in a program to have a certain effect, or might use repetition to accomplish the same task in a smaller number of steps. Students will have the opportunity to learn, practice, and apply the coding concepts through unplugged activities and guided coding activities. Then, students will have the opportunity to demonstrate their new knowledge and skills in the independent project.

LESSON OVERVIEW

All of the following lessons are approximately 45-60 minutes, depending on how much time students are given to explore the coding activities:

- Lesson A: Introduction to Iteration
- Lesson B: Coding with Iteration
- Lesson C: Debugging with Iteration
- Lesson D: Get Creative with Iterations

LESSON OBJECTIVES

- Learn and apply the coding concept of iteration
- Explain the use of iteration in coding and synonymous terms (i.e., loops)
- Describe the different kinds of loops in MakeCode
- Use different types of loops to code and debug more efficiently
- Design an original project to apply the use of loops to create an automated solution

THINGS TO KEEP IN MIND

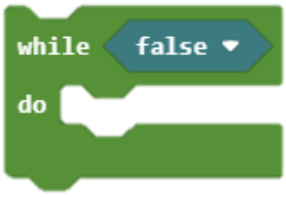
- You will need to gather the additional materials for the unplugged activities:
- Remind students that there may be more than one solution for each of the activities.
- Students should keep a coding mindset. As with any other sport or activity, you must practice in order to get better. Coding is no different.

MINECRAFT MECHANICS

C	**C** Summons the Agent and opens the MakeCode interface
T	**T**

	Opens chat panel in Minecraft for commands to be typed
ESC	**ESC** When a student wants to leave the game, leave chat, or pause the game

CODING BLOCKS

	**Repeat** Run part of the program the number of times you say.
	**For** Run part of the program the number of times you say using an index variable.
	**While** Repeat code while a Boolean condition is true/false.
	**Forever** Run a part of the program in the background and keep running it over again.

KEY VOCABULARY

Iteration – process wherein a set of instructions or structures are repeated in a sequence a specified number of times or until a condition is met
Loop – structure to repeat code for a specified number of times

LESSON ACTIVITIES

Lesson A: Introduction to Iteration (Slides 1-32)

Direct Instruction (Teacher-Led; "I Do")

Welcome to back to Coding with Minecraft, a place to learn all about the basic concepts of computer science. Over the next couple of sessions, we are going to learn about iteration. (slide 1)

Today, we are going to learn about what is iteration and how iteration is applicable in real-life, but how they are used in Minecraft. (slide 2)

Review the lesson objectives. (slide 3)

So exactly what is a loop? (slide 4)
In programming, there are some situations in which you want to do the same thing many times.

For instance, in Minecraft you might want to place 10 grass blocks. A loop is a block structure that allows you to easily repeat placing a grass block. In the Block editor, loops are green. There are different types of loops for different situations.

You use loops to repeat code as many times as you want. When programmers talk about repeating code, they use many other words as well. All these words basically mean "repeat." You will find them used, at times interchangeably, when you are learning about code. (slide 5)

- iterate
- iteration
- iterative
- repeat
- repetition
- repetitive
- loop

Programmers use sentences like this: "Let's iterate through the loop." This means, "Run the same code many times, repeating something."

Walking is iteration… walking is repeating… walking is an action we do in a loop…
We don't think much about it, but our bodies repeat many actions. Breathing, sleeping, eating, drinking, and walking are all actions that are repeated daily. Walking breaks down to repeating these steps:

```
.----.
|    |
|    v
|  Step 1 - "left foot forward"
|    |
|    v
|  Step 2 - "right foot forward"
|    |
|    v
'- REPEAT
```

Other examples

What are some other examples of iteration or repeating in your daily life? What tasks and actions are composed of repeated steps? What are those steps? (**slide 6**)

- Search the web for Ancient Roman aqueducts - Architects use repetition in their building designs.
- Search the web for dog fetching ball on beach - Dogs love to play fetch as a repetitive activity.
- Search the web for honeycomb - There are many repeated patterns found in nature.

Guided Instruction (Teacher Modeling; "We Do")

Unplugged Activity: Walk Around the House (slide 7)

Materials Needed: Chairs (or other objects)—number will depend on the number of pairs

This activity works best with a partner. You will write instructions for your partner to walk completely around the outside of a Minecraft house and return to the start. You will need a chair and some way to write things down. Place the chair in the center of the room. The chair represents a Minecraft house. Your partner should stand next to one of the corners of the house.

Your goal is to write the instructions for your partner to walk all the way around the house, using these two commands: (**slide 8**)

```
forward()
turn left/right()
```

Your pseudocode will probably end up looking something like this:

```
forward()
turn left()
```

```
forward()
turn left()
forward()
turn left()
forward()
turn left()
```

Have your partner follow your algorithm to prove that it works.
Now notice that this is eight lines of code, and many of the lines repeat. In particular, the following two lines of code repeat four times:

```
forward()
turn left()
```

Programmers like to take shortcuts. A program that has fewer lines of code takes up less room in memory, and ultimately the shorter your program is, the easier it is to find mistakes. Whenever you have code that repeats, you have an opportunity to use a loop to simplify your code.

How could this be shortened? How about:

```
Repeat 4 times:
   forward()
   turn left()
```

Go ahead and follow this revised code to prove that it works.

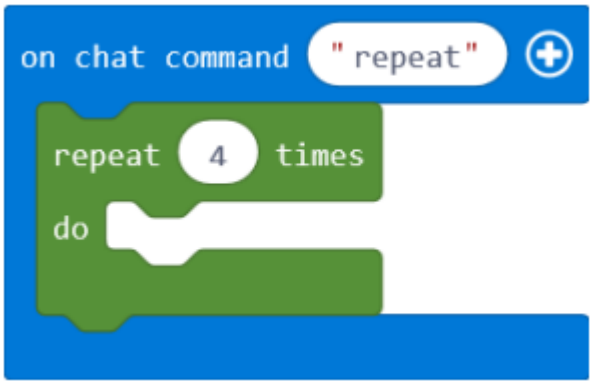
You have just rewritten eight lines of code as three lines of code by using a loop. The “repeat” command creates a loop. The code within the loop gets repeated a certain number of times, and then the program exits the loop.

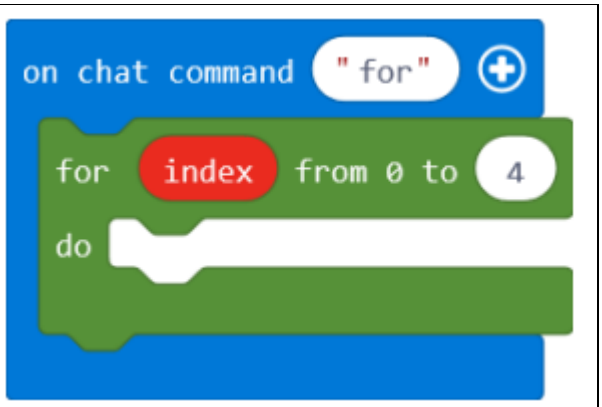
Types of loops in MakeCode (slide 9)

Computers use loops to repeat sets of instructions under different conditions. Here are some examples of loops you will find in MakeCode. See if you can think of some different types of activities that would be appropriate for each type of loop.

Counted Loops

Repeat *N* times	For Index 0 to *N*

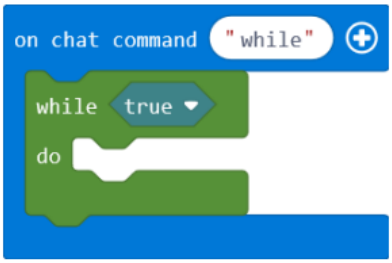




Both repeat and for loops repeat _n_ number of times. If you have 32 stairs in your house, going upstairs or downstairs would probably involve taking one step 32 times. Other examples of activities that repeat a set number of times include entering a building with double front doors, lacing your shoes, and turning the pages in a picture book.

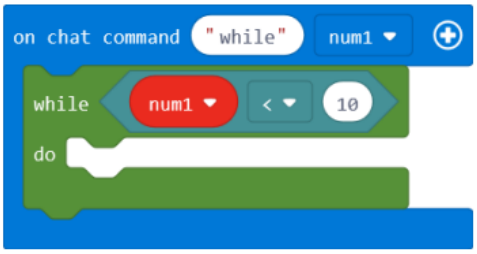
Similar to a repeat block, a for block also performs its actions the number of times indicated, but it uses a variable called index that you can use inside the loop to check (although index is used by default, you can change this variable). If you need to do something on the fifth time through a loop, this might be a good loop for you!

While (true) loops



A while loop runs as long as the condition inside of it is true. By default, true is always true, so in effect, a while <true> loop runs forever. Usually, though, you replace the <true> with a condition that evaluates to true or false. For example, while <hair is messy> brush hair would continue to repeat brushing hair until <hair is messy> is false. Some examples of activities that might use a while loop include while <soup remains> eat with spoon; while <energy greater than 0> do jumping jacks; and while <she hasn't seen me> wave furiously.

A while loop is a good loop to use with an if (conditional) statement inside.



In Minecraft, you might have a while block that repeats as long as the Agent detects redstone beneath it. You can think of a while block as a repeat block combined with an if conditional statement – if there's redstone beneath me, do these things.

Forever loops



A forever loop starts running when the program starts and keeps going until the program ends. Some real-life examples are breathing or the beating of your heart.

Before moving on with the lesson, briefly check for understanding with the following questions: (**slide 10**)

- What's another word for iteration?
- What are two ways that loops are beneficial when coding?

Coding Activity: Introduction to the Agent (slide 11)

Before you begin this coding activity, make sure you are set-up to demonstrate and lead the students through this coding experience. It is best if you complete this activity on a device connected to a projector or presentation screen. Split the screen and display the classroom presentation alongside of the Minecraft world.

Have all students log into Minecraft: Education Edition. After students have logged into the platform, navigate to the Coding with Minecraft subject kit. (**slide 12**) Once in the Coding with Minecraft subject kit, select Unit 7: Iteration to begin the lesson. (**slide 13**)

For this coding activity about loops, you will be using the Agent in Minecraft. The Minecraft Agent is a little robot who can carry out the commands you write in MakeCode. This activity will walk you through getting started with the agent. (**slide 14**)

Once inside of the world, press “C” to open Code Builder. Start a new project and name it “Agent Moves”. (**slide 15**)

Steps for the Coding Activity: Agent Moves (slides 16-18)

- **Step 1:** Change the name of the default on chat command block to "**tp**".
- **Step 2:** From the AGENT Toolbox drawer, drag an agent teleport to player block and drop it inside the on chat command block.
- Create some additional basic commands to move the Agent around: forward, back, right, left.

Students should have the time to play around with the Agent to see what else it can do! (**slide 19**)

Coding Solution:



Challenge 1: Make the Agent Fly

The Agent can move up and down as well as forward and backward.

Create some additional on chat command commands that will move the agent up and down.

Challenge 2: Move Your Agent a Set Distance

Let’s give the agent very specific directions for movement. How would you make your agent move forward four spaces, turn right, and then move backward six spaces?

You could do this in many ways. Regardless of how you solve this challenge, it might be easiest to create a new on chat command block and put your solution there.

Extensions

There are no rules for these extensions. See what you can come up with!

Extension 1- Fire Walk

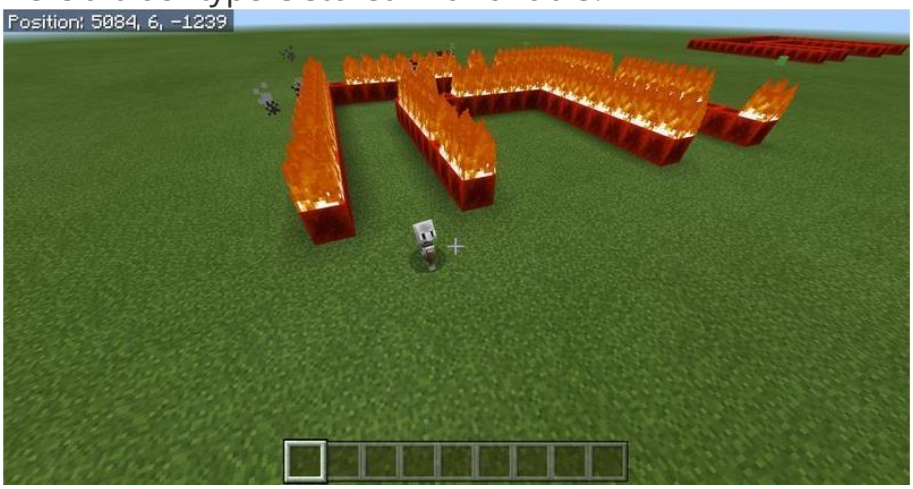
This code will create a fire course for your agent to walk through. Then you can use the on chat commands to make the agent move through the course. The movement commands from before have been modified a little so that you can now put numbers into your commands.

Using the code

1. To get started, enter **course** in the chat window. This creates your fire course.
2. Next, enter **fd 5**, for example, to move your agent forward five blocks. The around command spins the agent 180°.

How might you modify this code? Could you change what the course is made out of? Could you add a run-and-jump command for your agent? Could you make a course that requires the agent to move up and down?

Notice that here a block type is stored in a variable!

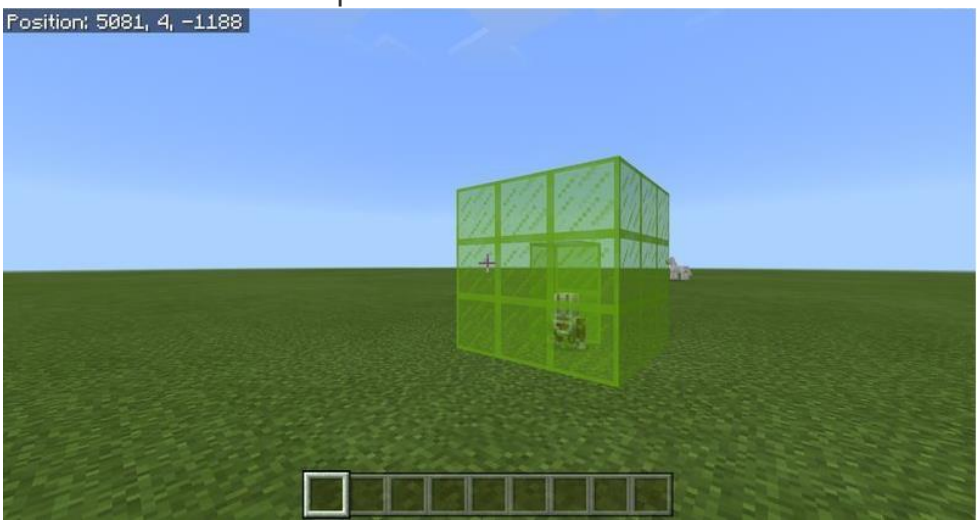


Extension 2 – You Can't Contain the Agent!

This code builds a cage for the agent. Enter **contain** to build a cage, teleport your character into the cage, and teleport the agent into the cage with you. When you teleport out, you have trapped the agent! Or have you? Everybody knows you can't really trap an agent.

The break command frees the agent!

How could you modify this? It would be more impressive if the agent destroyed the whole cage. You could teleport the agent underground and have it dig back out. There are tons of possibilities.



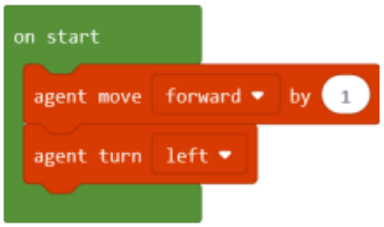
Independent Work (Teacher Support; "You Do")

Coding Activity: Dance Dance Agent

In this activity, you'll create a unique dance for the agent to do, using repeat loops and for loops. First, you will make one full sequence of moves, and then use the loops to repeat your dance as many times as you want.

Dancing is a perfect example for loops because when you dance, you often repeat the same moves.

In the AGENT Toolbox, you will see the following commands:



You can select a different direction for each of these commands by choosing from the drop-down menu. The agent move command moves the agent in the direction the agent is facing.

agent turn turns the agent. This can be used multiple times to spin the agent around 360 degrees.

There are also additional actions for the agent. At the end or start of these actions, the agent does an animation. These might be interesting to add to your dance.

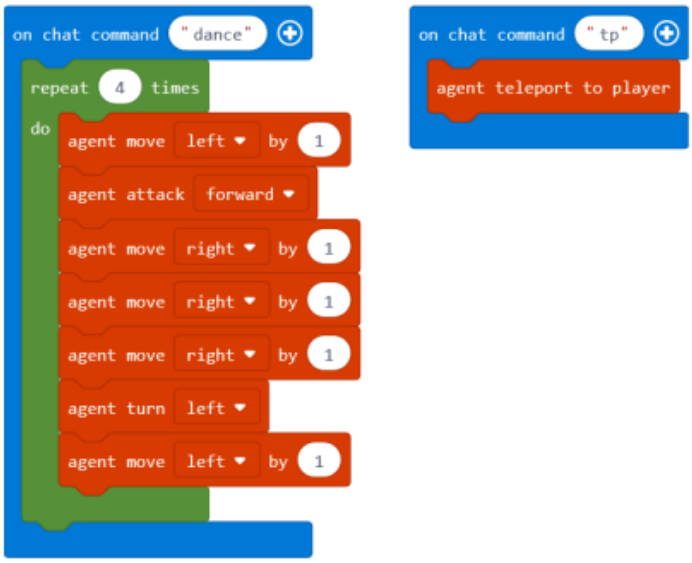


You have the moves; now you just need to mix them together! Feel free to mix some of those commands in to make your dance more interesting.

Steps for the Coding Activity:

1. Name this command "dance".
2. Create a sequence of moves for your agent to follow.
3. From the LOOPS Toolbox drawer, drag a repeat block into the on chat command "dance" block and surround the agent blocks.
4. Use the repeat block to repeat your sequence of dance moves as many times as you want.
5. From PLAYER, drag a new on chat command block to the coding Workspace.
6. Name this command "tp".
7. From AGENT, put agent teleport to player inside on chat command "tp".

Sample Code:



Challenges

Challenge 1 - A Second Repeat Loop

Recall that LOOPS are great for things that repeat. In the dance above, you use one repeat loop, but actually there is a place in the code where you could use a second repeat loop.

You would repeat this new loop three times.

Challenge 2 - Adding a 3, 2, 1 Start

This challenge will test your learning of variables and loops.
Your challenge is to create a loop that will display a 3, 2, 1 message before the robot starts dancing.
To do this, you would need to:

1. Create a variable that you will change from 3 to 2 to 1.
2. Use a loop.
3. Use say.

The say block will accept only strings, but your variable will be a number. You need to cast (change) your number into a string. If you don't, you will get an error.

Therefore, use `join` like this:



Extensions

Here there are no rules... Copy the code for the experiments and change things around to see what kind of results you can create. Suggestions are given, but do as you like!

Extension 1 - You Dance When I Say

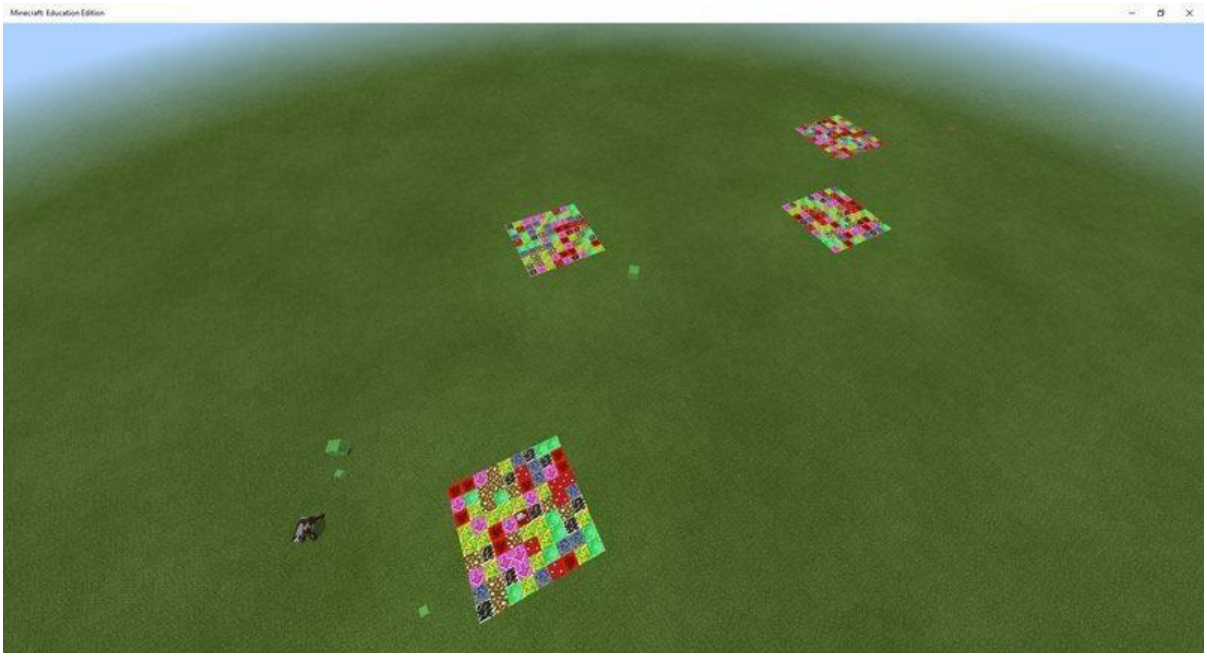
In this code, you use different kinds of loops to stop and start the robot dancing whenever you want. You might use the same structure to start the robot digging, collecting, or doing anything you want. Give it a try!

Extension 2 - Instant Dance Floor

This code uses two functions to make a random dance floor. You will also use a bit of teleporting trickery. First, you teleport your character, and then you teleport the agent. Finally, you teleport your character back to its original spot.
How could you change this code? How could you use the two functions that make random blocks in a different way? Could you add even more to set the mood for a dance contest? Night time? Lights?

Could you find a way to keep the robot on the dance floor no matter what dance moves he does?





LESSON CONCLUSION

Upon completion of this lesson, students should be able to answer the following questions: (**slide 22**)

1. Are agent commands based on coordinates or the direction the agent faces?
   Answer: Agent direction

2. What another word for iteration?
   Answer: Any one of the following: repeat, repetition, repetitive, loop, iterative, iterate

The following exit-ticket question can also be used as a formative assessment. A printable version of these questions can be found in the Assessment Guide.

1. Which of these words cannot be used interchangeably with iteration: repeat, rendition, repetitive, loop, iterative, iterate?
   Answer: Rendition
2. What's a task you could teach your Agent to do?
   Answer: Responses will vary

Conclude the lesson by reviewing the concepts covered in Lesson A. (**slide 23**)

Provide a preview of the upcoming lesson. (**slide 24**)

Lesson B: Coding with Iteration (Slides 25-45)

Direct Instruction (Teacher-Led; "I Do")

Welcome back to Coding with Minecraft. We are going to continue to learn about and practice using iteration in our coding. (**slide 25**)

Review the lesson objectives. (**slide 26**)

Unplugged Activity: Everyday Tasks (slide 27)

Have students think of a task and then, on a sheet of paper, have them describe the repeated steps that comprise this task.
(Display **Slide 28**).

1. Pull your __ from your pocket.
2. Press the power button on your __.
3. Unlock it.
4. Get onto the internet.

After everyone has their steps written out, have the students exchange steps. After students have exchanged their steps, someone might correctly guess that the repeated action just described is someone using their cell phone.

Is this a repeated action? If so, it would be good to use a loop for this if it were in code. You might even draw a picture of a loop block and write the name of an example task in the middle to get students to bridge the connection to coding. If you put these steps in the loop in the picture, how many times would you repeat the steps?

Guided Instruction (Teacher Modeling; "We Do")

Coding Activity: One Block at a Time
Import project from here:
https://minecraft.makecode.com/?ipc=1#tutorial:github:mojang/educationcontent/CodingWithMinecraft/unit_7/lesson_C

Make sure you are set-up to demonstrate and lead the students through this coding experience. It is best if you complete this activity on a device connected to a projector or presentation screen. Split the screen and display the classroom presentation alongside of the Minecraft world.

Have all students log into Minecraft: Education Edition. Students should continue working in the same world from Lesson A. Directions for navigating to the previous world can be found on **slide 30**.

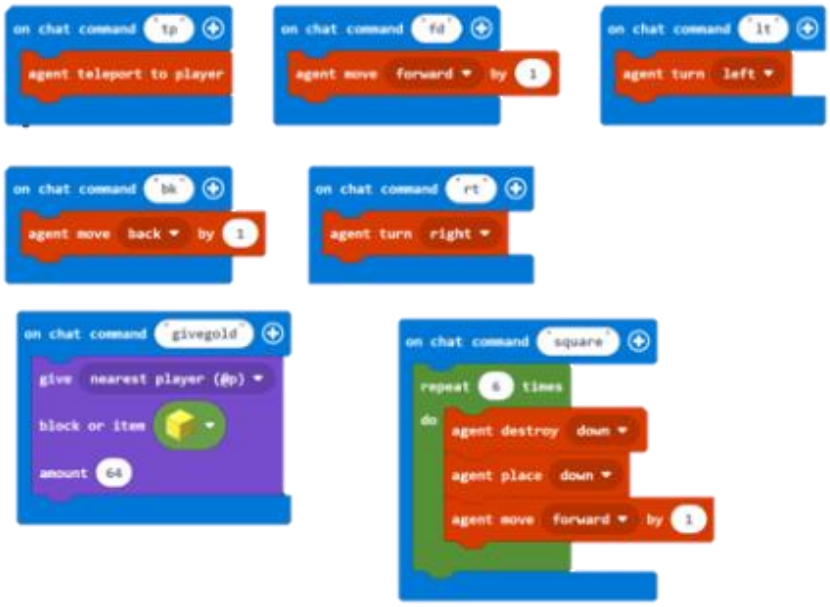
Use this starter code from the Introduction to the Agent activity. This will allow you to easily position your agent, and then you can adapt some of it to do something new. (**slide 30**)

Steps for One Block at a Time: (slides 32-37)

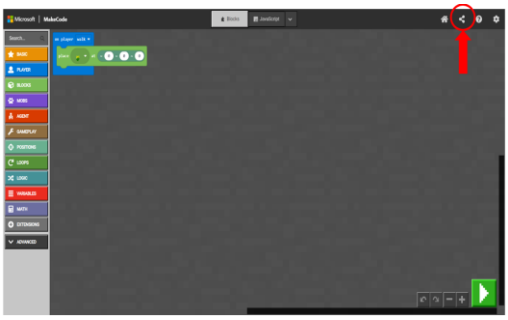
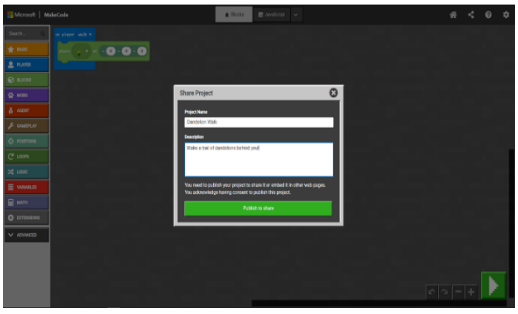
- Use the starter code from the Agent Moves activity. This will allow you to easily position your agent, and then you can adapt some of it to do something new.
- Rename the command to **givegold**. Go to the Search and find give.
- Give yourself blocks, and then equip your agent manually. Change the 'Give' so you are giving your character blocks of gold. The maximum you can hold in one inventory slot is 64, so give your character 64 blocks of gold.
- Now that you have blocks of gold equipped, you need to equip your Agent. Use the on chat command 'tp' to teleport your agent near you or walk over to your Agent.
- Right-click your agent to see the Agent's inventory and your character's inventory. Transfer the blocks of gold to your Agent.

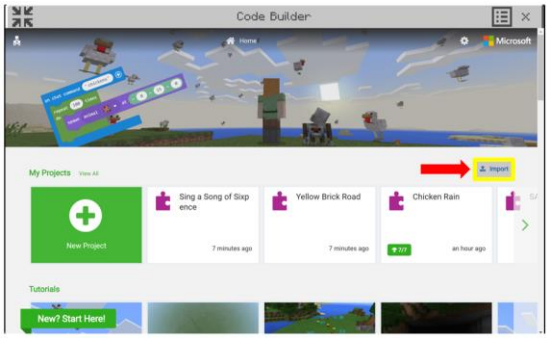
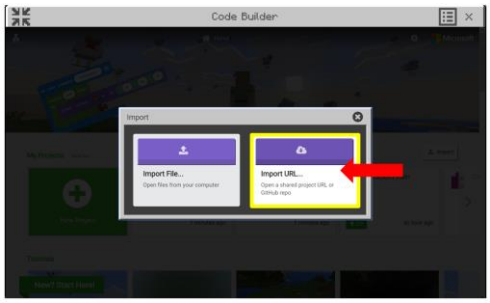
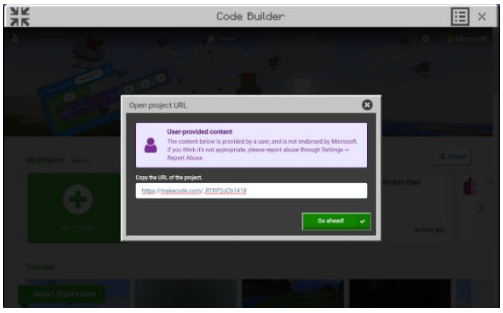
- From AGENT, put agent destroy down inside on chat command "square".
- Put agent place down inside on chat command **"square".**
- From LOOPS, grab a repeat loop to surround the existing code in on chat command “square".
- Adjust the repeat loop to repeat six times.
- Adding agent move forward by 1 at the end of this loop will get your agent moving while he destroys and places blocks.

Coding Solution:



Then, you will need to publish and import the program into Code Builder.

Step 1: Share your program.	**Step 2**: Name the project, provide a brief description, and select the “Publish to share” button.
	

Step 3: Select the green “Copy link” button.  NOTE: If you are having difficulty with the copy link, try using the short cut: press Ctrl + C (or Command + C) on your device.	**Step 4**: Inside Code Builder, select the “Import” button. 
Step 5: Select the “Import URL” button. 	**Step 6**: Paste the Project URL. Then, select the “Go ahead!” button.  NOTE: If you used the Ctrl + C (or Command + C for a Mac) to copy the link, you would then paste by pressing Ctrl + V (or Command + V) to paste.

Now, we are going to return to our Minecraft world to try out our code. **(slide 38)**

Independent Work (Teacher Support; “You Do”)

Now we need to repeat this line four times. Can you figure out how to accomplish what’s shown in the following picture?



Challenge 1 - Turn One Line into a Square

What is a square? In this case, let's say four lines. How do you make a square by using the code you have up to this point as a starting place?
The lesson is about loops, so you might need another, but you will also need one other block.

Challenge 2 - Make a Tower

A tower is just several squares laid on top of each other. Can you add yet a third loop to accomplish this?



What happens when you try to build a gigantic tower?



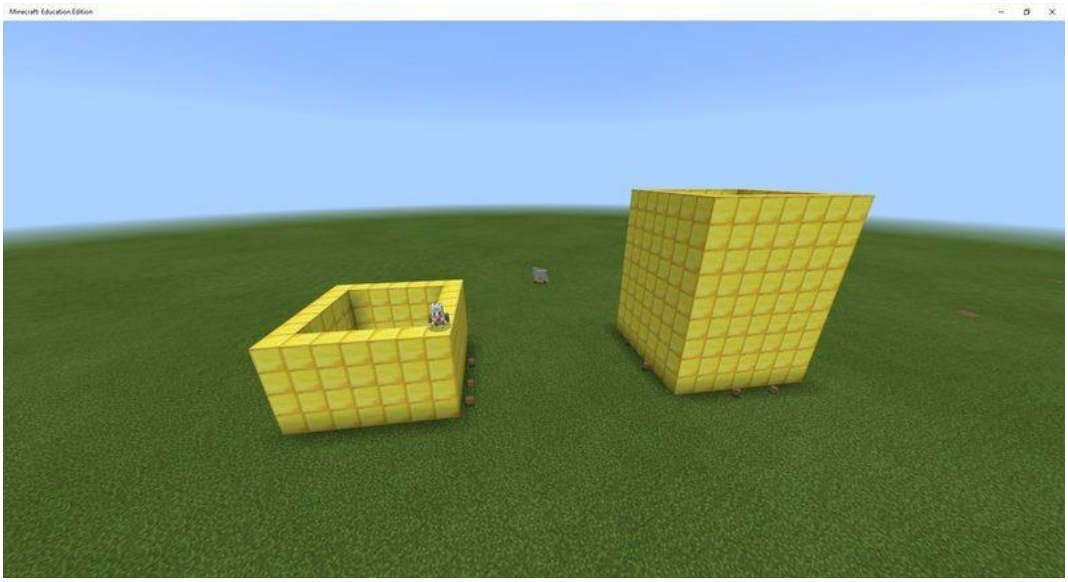
Extensions

Here, there are no rules... Copy the code for the experiments and change things around to see what kind of results you can create. Suggestions are given, but do as you like!

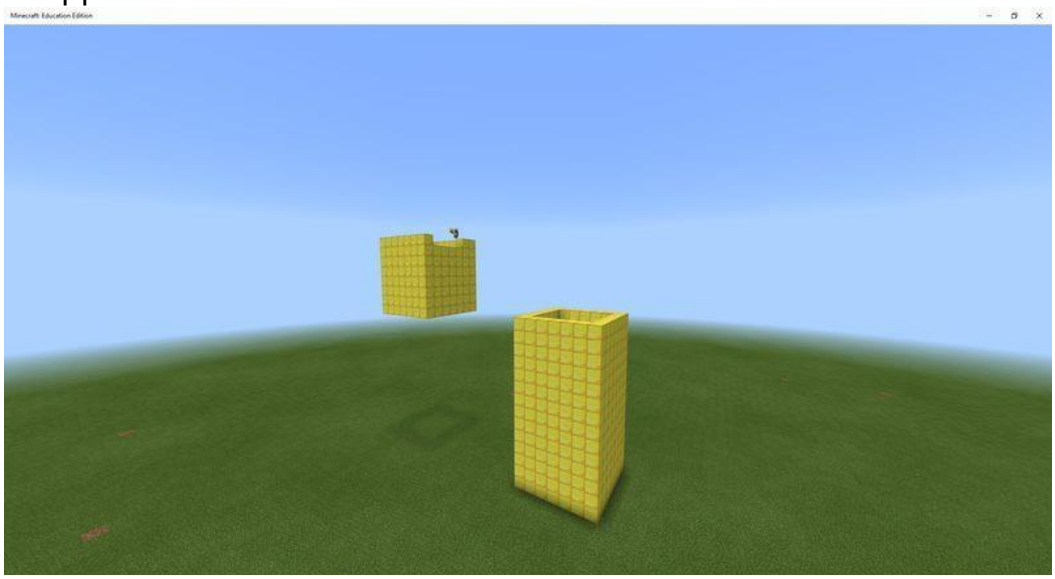
Experiment 1 - Starter Castle

How far can you go with a loop inside a loop inside a... you get the idea. These are called nested loops, and if you add another with some code, can you make a castle?

Most castles have four towers. With this code you get two. Go get a sandwich while your Agent does all the work. After this, you can write code to make it build the other two towers and walls!



What other weird tower designs could you think up? Tinker with the code and make it happen!



LESSON CONCLUSION

Upon completion of this lesson, students should be able to answer the following questions: (**slide 39**)

1. When an agent places blocks, which corner of his inventory is used?
   Answer: Whatever is in the upper-left corner
2. How do you see your agent's inventory?
   Answer: Point at the Agent and right-click or tap and hold

The following questions can also be used as an exit ticket (formative assessment). A printable version of these questions can be found in the Assessment Guide.

1. What type of loop does this describe: while <soup remains> eat with a spoon?
   Answer: A while loop
2. What kind of loop starts when a program starts and keeps going until the program ends?
   Answer: A forever loop

Conclude the lesson by reviewing the concepts covered in Lesson B. (**slide 40**)

Provide a preview of the upcoming lesson. (**slide 41**)

Lesson C: Debugging with Iteration (Slides 42-66)

Direct Instruction (Teacher-Led; "I Do")

Welcome back to Coding with Minecraft. We are going to continue to learn about and practice using iteration. (**slide 42**)

Review the lesson objective. (**slide 43**)

Guided Instruction (Teacher Modeling; "We Do")

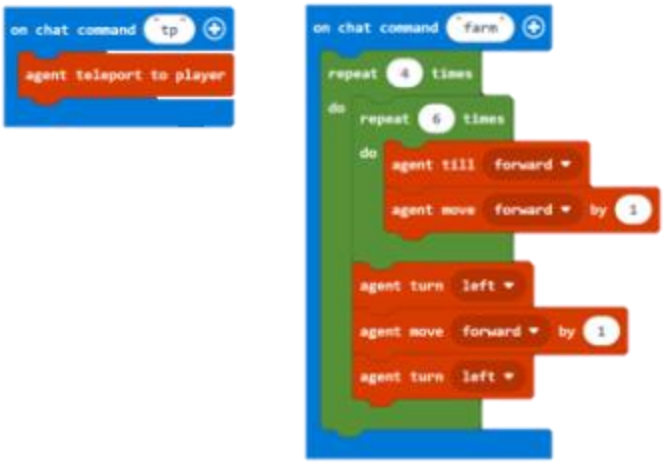
Before you begin this coding activity, make sure you are set-up to demonstrate and lead the students through this coding experience. It is best if you complete this activity on a device connected to a projector or presentation screen. Split the screen and display the classroom presentation alongside of the Minecraft world.

Have all students log into Minecraft: Education Edition. Students should continue working in the same world from Lesson B. Directions for navigating to the previous world can be found on **slide 44**.

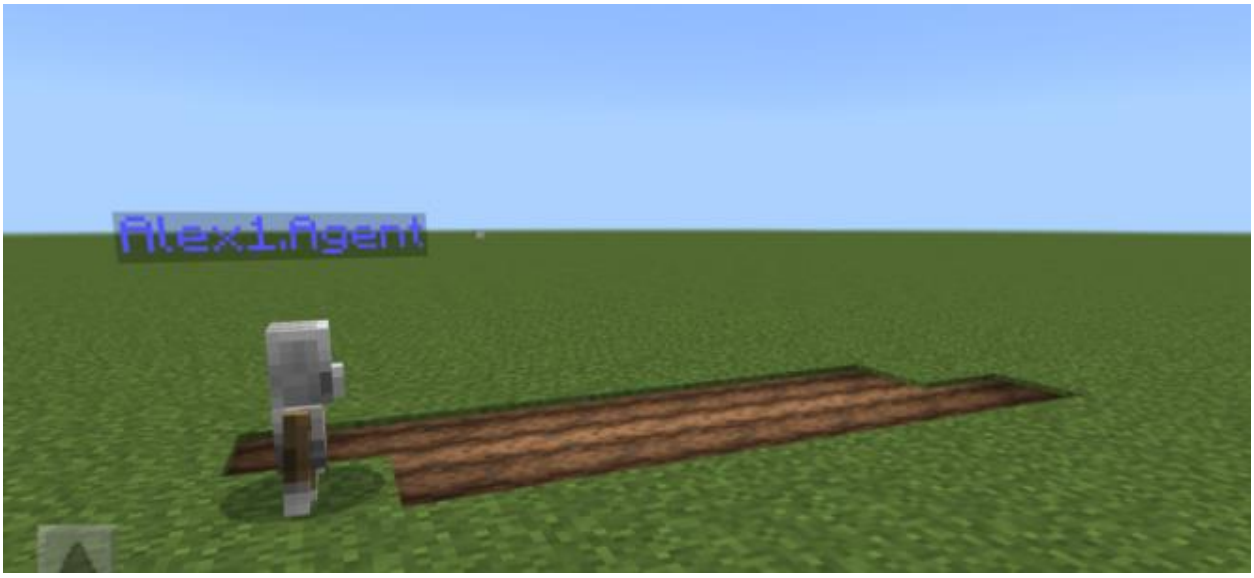
Coding Activity: Help Agent Farmer (slide 45)
Tutorial can be found here:
https://minecraft.makecode.com/?ipc=1#tutorial:github:mojang/educationcontent/CodingWithMinecraft/unit_6/lesson_C

Steps for Coding Activity (slides 46-50)

1. By default, new projects start with a on chat command, but you will need two. Duplicate this on chat command by right-clicking and selecting duplicate.
2. Name one "tp" to teleport the agent to your location, and name the other one "farm".
3. From AGENT, put agent teleport to player inside on chat command "tp".
5. From LOOPS, drag a repeat block into on chat command "farm".
6. This will determine the length of our row, so in repeat change 4 to 6.
7. From AGENT, put an agent till and agent move block into repeat.



So far, using the current code, the agent will till the soil directly in front of it (with its own diamond hoe) and move forward into that space. It does this six times because of the repeat loop. Try running the code in Minecraft to see what happens.



Notice where the agent ends up after the repeat block finishes. This is great, but you need two rows! You need to get the agent to turn around and till the next row over so it ends up like the following picture.



Independent Work (Teacher Support; "You Do")

Students should work on debugging the problematic code. (**slide 51**)

Coding Solution:



Challenges

Challenge - Get Agent to Till a 5x6 Plot of Aligned Soil

Can you change the dimensions of the garden? We need more carrots! Try to make a 5x6 garden instead of 2x6.

There are many ways to do this. Can you figure out a way to do it by adding a third loop? If not, then try another approach. Positioning your agent before each loop starts is the most important thing to consider.



Extensions

Here, there are no rules... Copy the code for the experiments and change things around to see what kind of results you can create. Suggestions are given but do as you like!

Experiment 1 - Creating a Help Menu for Users

Controlling the agent is not easy, and sometimes it is easy to forget all the commands. This code has a built-in Help menu. It takes in commands as integers from 1-6. Entering a '?' gets you help or you can just enter do without an integer to get the Help menu.

How can you change this code so it moves the agent forward based on a number the user enters for the on chat command "do"? What if you wanted to enter something like 'do 1 4', for example, to move your agent forward four? Is this possible? What other code could you build a Help menu for?

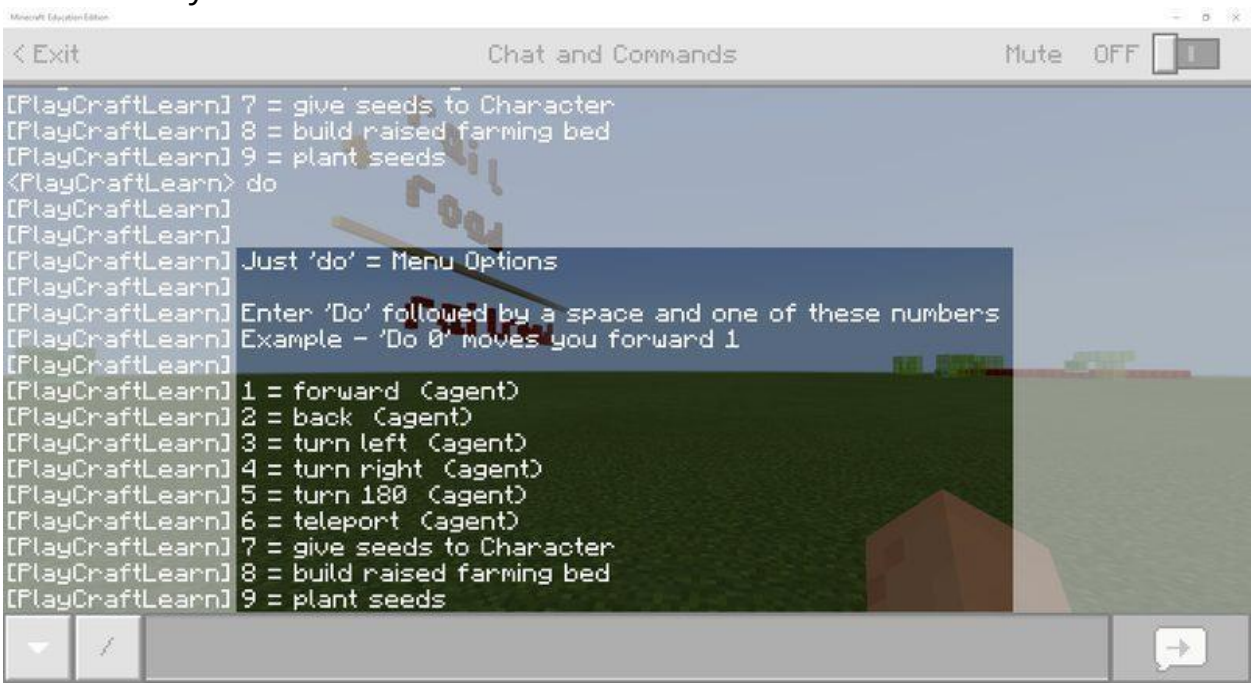


Experiment 2 - Creating a Raised Farming Bed with Planted Crops

You can use this code to create a raised farming bed with crops. This is a huge amount of code, but if you look at it block by block, you can probably figure out how it works. You use functions to organize your code a bit.

Also, you use the builder blocks to construct your raised farming bed. Feel free to play around with these.

Here is how you use this code:



1. Choose a nice spot for a raised farming bed.

2. Teleport your agent to your character's location (chat command 'do 6').
3. Give your character some seeds (chat command 'do 7').
4. You will need to transfer seeds to the agent's inventory so the robot can access them when it plants. To do this, right-click on your agent. Move your character's seeds to your agent's inventory as shown in the following picture. The seeds must be put in that upper-left box, or the code will not work.



5. Build the raised farming bed (chat command 'do 8').
6. Now you will need to turn your agent so that it is facing you. If you don't turn it correctly, the agent will not plant where you want. Turn the agent to face you as shown in the picture. Chat commands 3-5 allow you to turn the agent easily.



7. Plant (chat command ‘do 9’)!



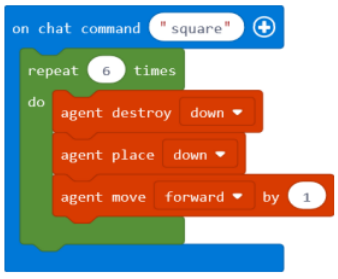
How would you modify this to make your farming bed bigger? Could you get your agent to plant different crops? What about putting plants in some rows, water in others, and decorative blocks as a third style for rows? What about getting the agent to plant seeds in all the soil in the raised farming bed?

LESSON CONCLUSION

Upon completion of this lesson, review the concepts covered in this lesson. **(slide 52)**

Then, students should take the quiz (found in the assessment guide). This is intended as a low-stakes formative assessment tool to establish whether students are prepared to take on the unit independent project.

<table>
<tr><th>Questions</th><th>Answers</th></tr>
<tr><td>1. What's a benefit of iteration in coding?</td><td>Any of the following:
Fewer lines of code
Simplified code
Takes up less room in memory
Shorter programs
Makes it easier to find mistakes</td></tr>
<tr><td colspan="2">2. Match the following types of loops with their definition:

Repeat → Runs a command as long as a condition is true
For → Runs a command n times
While → Runs a command until the program ends
Forever → Runs a command n times with a variable</td></tr>
<tr><td>3. Is the following statement TRUE or FALSE:
The Agent will place whatever is in the lower-right corner of his inventory.</td><td>False</td></tr>
<tr><td>4. Both functions and loops can be used to repeat code. What's the difference in how they do this?</td><td>Loops repeat code over and over in one specific place.
Functions spread repeating code throughout applications.</td></tr>
<tr><td>5. Describe what the following coding blocks do with words:

</td><td>Something like:
When you type "square" in the chat command field
The Agent will destroy the block below it
Place a block in that space
And move forward by one block
To create a line of six blocks, like a trail or road</td></tr>
</table>

Then provide students with a preview of the next lesson. **(slide 53)**

Lesson D: Get Creative with Iteration (Slides 54-67)

Direct Instruction (Teacher-Led; "I Do")

Welcome back to Coding with Minecraft. Today, we are going to have the opportunity to demonstrate our new knowledge and skills about iteration. (**slide 54**)

Review the lesson objective. (**slide 55**)

In this unit, you learned how organizing blocks of code into separate functions makes your code more readable and saves space. Functions generally contain sections of code that go together logically and accomplish one thing. The function's name usually tells you what it does.

Functions can call other functions. Although functions in MakeCode do not accept parameters, you can use an on chat command block to pass in a parameter. You might pass in a number to affect what the function does. Also, you could update the value of an existing variable and then any function can access that.

Guided Instruction (Teacher Modeling; "We Do")

The project for iteration is a bit different from previous projects. This independent project is an opportunity for you to show what you know. In this lesson, you learned about how LOOPS can reduce the size of your code by repeating certain tasks for you. Now, your challenge is to come up with a MakeCode project that uses iteration in some way to create a stairway to diamonds.

One of the most valuable resources in Minecraft are diamonds. You can find diamonds deep underground in **Survival** mode. Getting from the surface down to where diamonds are found is challenging. Luckily, you can program your agent to do this for you! For this project, come up with a way to teach your agent to dig a tunnel down to around layer 10-13 underground, where diamonds are usually found. After you are at that level, you can dig parallel tunnels in search of diamonds, or even program the agent to do it for you. **(slide 56)**

For this independent project, choose one of the following options. You can build any of these styles of staircases, but remember to use loops in your code to help do it. (**slide 57**)

Option 1 - Straight Staircase (slide 58)

A straight staircase is most common, and it has the added benefit of allowing you to run up and down quickly if you line the rock with stairs.

This challenge is a little easier if you aren't placing stairs, but there are a few things you should keep in mind.

1. You will need to jump up each level, and that increases the rate at which you need to eat.
2. Also, it's impractical to bring a horse down into the mine if you don't have a wide, tall staircase lined with stairs.





If you do decide to put down stairs…

1. By default, the agent places stairs facing backwards. This works well if you are building a staircase up, but if you are building a staircase down, think about how you could modify your code to make sure the stairs come out right.
2. If you try using the agent place on move block, the agent will place a block every time it moves when this is set to true. You may need to toggle agent place on move from true to false at certain points. Set it to true when you want to place stairs but false when you are just moving the agent and don't need to place anything.

Remember to use LOOPS to reduce repetition of code!
After you have the agent building a straight set of stairs, consider how you might modify your code to create a staircase three blocks wide and with enough headroom to allow you to ride a horse down the stairs!

Option 2 - Spiral Staircase (slide 59)

Another type of staircase is a spiral staircase. See if you can program the agent to dig this type of staircase for you.

		 Start by digging one block, then blocks, and then three blocks deep.	 After that, at each turn, remove a column of 3 blocks.
 You remove the block directly in front of the Agent. (Block 2 in this picture).	 You remove the blocks directly above and below that (Blocks 1 and 3 in the pictures).		Spiral Staircase EXAMPLE

A spiral staircase involves much of the same code that you would use to create a straight one-block-wide staircase. Think about where you might have to turn. And, as in all the staircases, you will probably need to follow the agent to place torches and clean up falling granite.

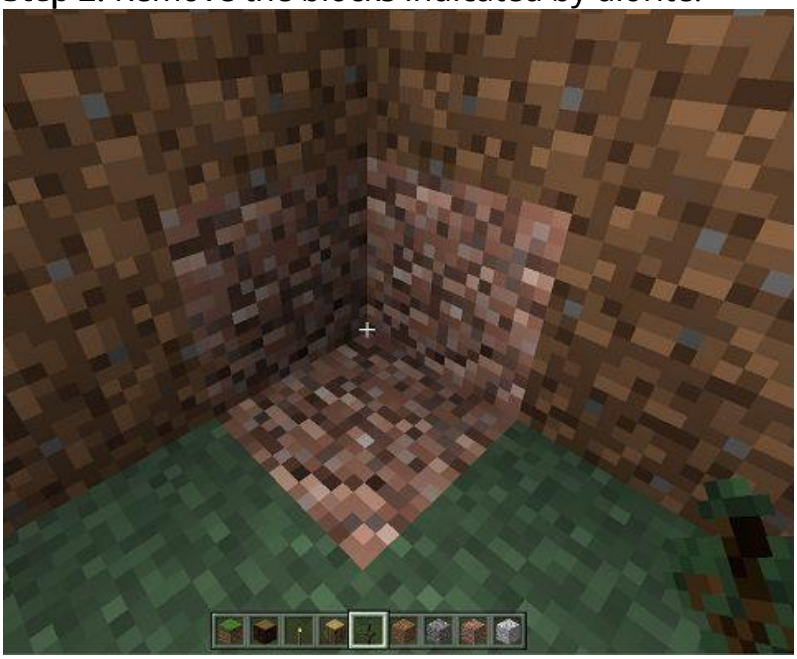
Option 3 - Diagonal Tunnel (slide 60)

One effective type of tunnel goes down at a diagonal angle. Can you create code that will get the agent to dig this type of tunnel? Digging a diagonal tunnel involves removing layers of blocks as shown.

Step 1: Remove the blocks indicated by granite.



Step 2: Remove the blocks indicated by diorite.



Step 3: Remove the center block.



Step 4: Continue in the same way, digging a diagonal tunnel.



Step 5: Tunnel is dug.



Note that the agent can't face diagonally like you can, so you will need to figure out how to move the agent so that it can dig in the right direction. Is this even possible?

Independent Work (Teacher Support; "You Do")

Coding activity: Independent project

Have all students log into Minecraft: Education Edition. Students are able to use any template/world they wish for this project. After students complete the independent project, they will be responsible for completing a Minecraft diary entry.

Minecraft Diary Expectations (slide 61)

Compose a diary entry addressing the following:

- What type of staircase did you choose to build: Straight, Spiral, or Diagonal? Why?
- What problems did you encounter? How did you solve them?
- How did you use loops in your staircase?
- Describe one point where you got stuck. Then discuss how you figured it out.
- Include at least one screenshot of your staircase.

- Share your project to the web and include the URL.

Grading Criteria
LOOPS
4 = Uses loops effectively in a way that is integral to the program
3 = Uses loops in a way that is integral to the program, but there are some problems with the loop output.
2 = Uses loops effectively but in a superficial way
1 = Doesn't use loops at all or uses loops in a superficial way and has problems with loop output

PROJECT
4 = Staircase is complete and navigable in both directions. Ends somewhere around layers 10-13.
3 = Staircase lacks one of the required elements.
2 = Staircase lacks two of the required elements.
1 = Staircase lacks all of the required elements.

LESSON CONCLUSION

Upon completion of this lesson, review the concepts covered in this lesson. (slide 62).

Preview the upcoming unit—Arrays. (slide 63)

EDUCATIONAL STANDARDS

CSTA Standards

- 2-AP-17 Systematically test and refine programs using a range of test cases.
- 2-AP-14 Create procedures with parameters to organize code and make it easier to reuse.
- 2-AP-11 Create clearly named variables that represent different data types and perform operations on their values.
- 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms.
- 2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
- 2-AP-19 Document programs in order to make them easier to follow, test, and debug.

ISTE Standards

- 1.4.c Students develop, test and refine prototypes as part of a cyclical design process.
- 1.5.a Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
- 1.5.c Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.
- 1.6.b Students create original works or responsibly repurpose or remix digital resources into new creations.

UK National Curriculum: Computing - Key stage 3

- Understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions
- Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits

Australian F-10 Curriculum: Digital Technologies - Year 7 and 8

- (ACTDIP027) Define and decompose real-world problems taking into account functional requirements and economic, environmental, social, technical and usability constraints
- (ACTDIP030) Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language
- (ACTDIP029) Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors