

MINECRAFT

EDUCATION EDITION



CODING WITH MINECRAFT 7: ITERATION

Student workbook

education.minecraft.net

CONTENTS

Overview.....	2
Unit summary	2
Learning goals.....	2
Coding concepts	2
Lesson A: Introduction to iteration	3
Lesson B: Coding with iteration	11
Lesson C: Debugging with iteration.....	16
Lesson D: Get creative with iteration	21
Glossary of key terms.....	26

OVERVIEW

Unit summary

In this unit, we'll explore ways to make things repeat. You might repeat actions in a program to have a certain effect, or you might use repetition to accomplish the same task in a smaller number of steps. We'll get to know the Agent, your own personal robot, who can accomplish tasks for you like building a farm, and we'll create some original dance moves, so your Agent can get down and dance on the floor. Finally, we'll challenge you to write your own program to direct your Agent to do something cool.

Learning goals

By the end of this unit, you'll be able to:

- Explain the use of iteration in coding and similar terms used by programmers.
- List examples of iteration in daily life.
- Describe the different types of loops in Microsoft MakeCode.
- Use different types of loops to debug and code more efficiently.
- Design an original creative project to automate a solution with iteration.

Coding concepts

Iteration: Code that tells a computer to REPEAT sets of instructions under different conditions.

Types of loop blocks in Microsoft MakeCode

Repeat n times	A counted loop that repeats an action n number of times.
For $index$ 0 to n	A counted loop that performs an action n number of times, using an $index$ variable. You can change it to another type of variable.
While ($true$)	A loop that runs as long as the conditional inside of it is $true$, like a repeat block combined with an IF conditional statement.
Forever	A loop that starts running when the program starts and keeps going until the program ends.

LESSON A: INTRODUCTION TO ITERATION

Overview: Iteration

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.

You can accomplish this by using “loop”, a block structure that allows you to easily repeat the code as many times as you want, like placing a grass block. 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.

- 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
```

Think about examples of iteration or repeating in your daily life:

- What tasks and actions are composed of repeated steps?
- What are those steps?

Write your ideas here:

You can search the web for other examples, like:

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

Today's activities

- Play an activity to get familiar with iteration
- Code your agent to dance

Unplugged activity: Walk Around the House

Imagine the classroom is a Minecraft game and that a chair is a house in the game.

Your goal is to write the instructions for a partner to start at one corner and walk all the way around the outside of the house back to the starting position, using these two commands:

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

Then have your partner follow your algorithm to walk around the house to prove that it works!

Write your pseudocode here:

How many of the lines of code repeat?

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 you rewrite your previous pseudocode with loops?

Write your shortened pseudocode here, then test it out:

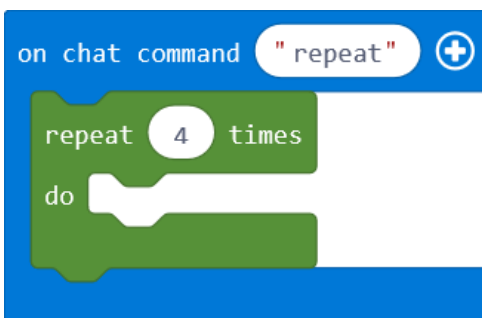
You have just rewritten eight lines of code as three lines of code by using a loop! The “repeat” command creates a loop. The code within the loop gets repeated a certain number of times, and then the program exits the loop.

Types of loops in MakeCode

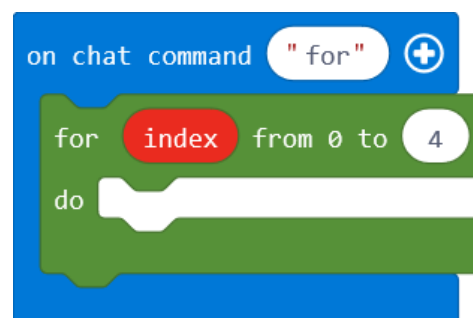
Computers use loops to repeat sets of instructions under different conditions. Here are some examples of loops you will find in MakeCode. See if you can think of some different types of activities that would be appropriate for each type of loop.

Counted loops

REPEAT n TIMES



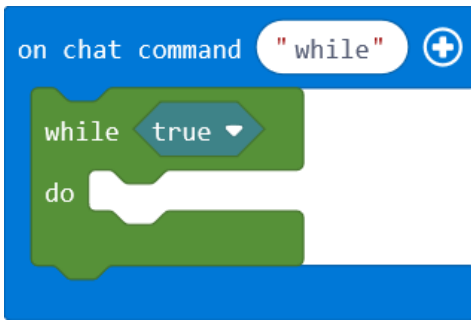
FOR INDEX 0 TO n



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

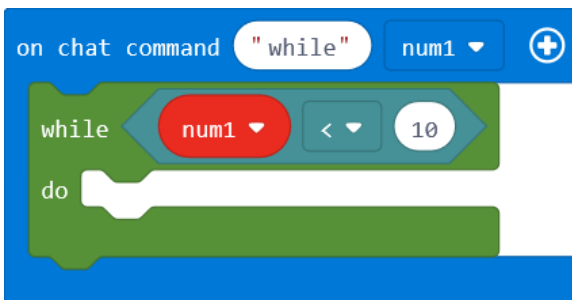


A **while** loop runs as long as the condition inside of it is true. By default, true is always true, so in effect, a **while <true>** loop runs forever. Usually, though, you replace the **<true>** with a condition that evaluates to **true** or **false**. For example: `while <hair is messy> brush hair` would continue to repeat brushing hair until `<hair is messy>` is false.

Other examples:

- `while <soup remains> eat with spoon`
- `while <energy greater than 0> do jumping jacks`
- `while <she hasn't seen me> wave furiously`

A **while** loop is a good loop to use with an **if** statement inside. Look for this when we talk about conditionals.



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.

Coding activity: Introduction to the Agent

The Minecraft agent is a little robot who can carry out the commands you write in MakeCode. In this activity, you'll use your agent to learn about loops.

Import the tutorial to MakeCode from here:

https://minecraft.makecode.com/?ipc=1#tutorial:github:mojang/educationcontent/CodingWithMinecraft/unit_7/lesson_A

Challenges

Now you can play around with the agent to see what else it can do!

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

Experiment 1 - Fire walk... I said NOW, Agent!

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. This was covered in Unit 4: Variables.

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!

Camera View

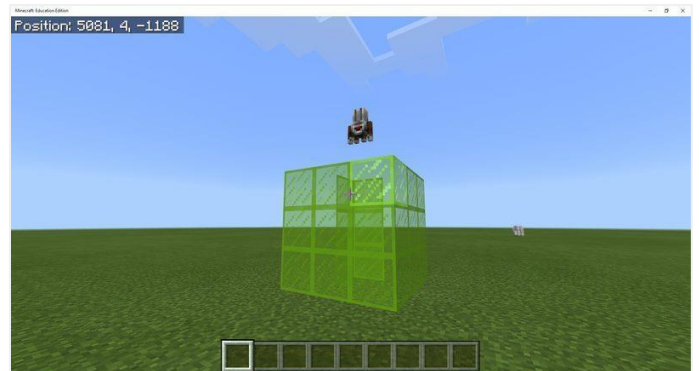
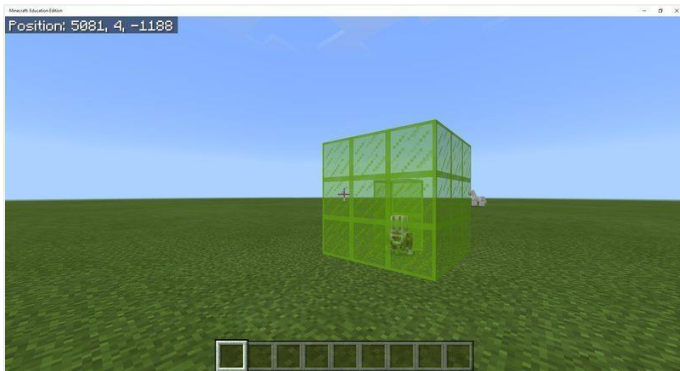
Pressing **F5** on your keyboard changes the view so you can see the agent and the fire course more clearly.



Experiment 2 - You can't contain this 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.



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.

Import the tutorial to MakeCode from here:

https://minecraft.makecode.com/?ipc=1#tutorial:github:mojang/educationcontent/CodingWithMinecraft/unit_7/lesson_B/dance

Challenges

Now you can change some things to make your own different and unique situations!

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.

Hint: 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. Hopefully, you can remember some of those concepts, but go back to review Lesson 4: Variables if needed.

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:

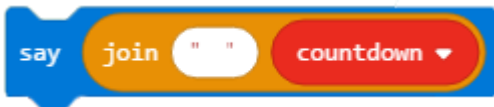
1. Create a variable that you will change from 3 to 2 to 1.
2. Use a **loop**.
3. Use **say**.

Hint: 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 this error:

Argument of type 'number' is not assignable to parameter of type 'string'.



Therefore, use **join** like this:



Experiments

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

Experiment 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!

Write your ideas here:

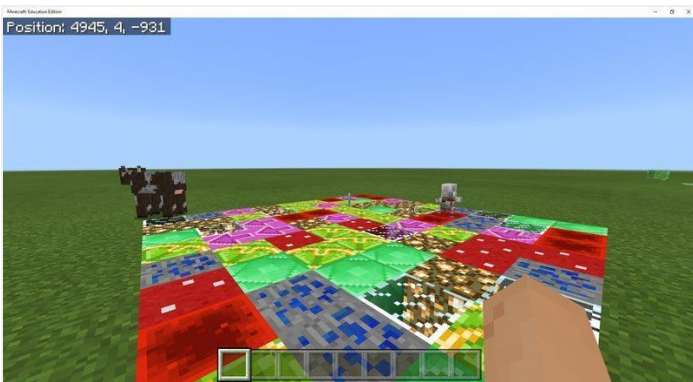


Experiment 2 - Instant Dance Floor

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 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?



Write your ideas here:

LESSON B: CODING WITH ITERATION

Today's activities

- Play an activity to identify iteration in everyday tasks
- Code your agent to build a tower

Unplugged activity: Everyday Tasks

Think of a task and then describe the repeated steps that comprise this task.

For example, here are some steps of a task. What's the repeated action?

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

Your turn: Think of a task.

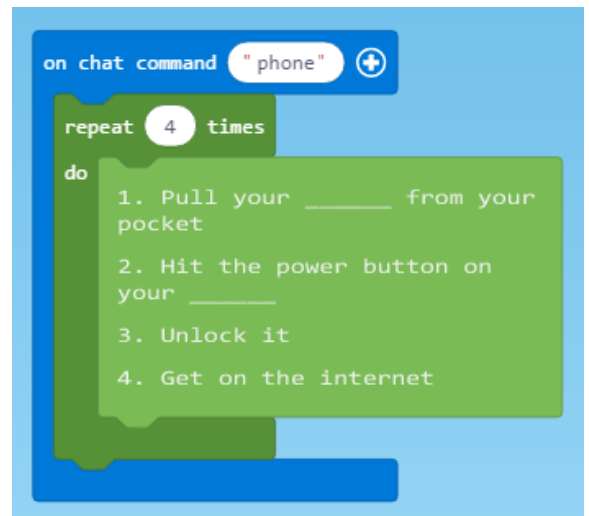
Write the repeated steps here:

Now, share your steps with someone and see if they can guess the repeated actions or task.

How could you use a loop if the steps were in code?

Remember, loops are ideal for repeated actions. So, it would be good to use a loop for this if it were in code. It could even be drawn as picture of the blocks.

A picture of the cell phone example might look like this:



Draw a picture of your task with blocks here:

Coding activity: One Block at a Time

This activity uses examples of looping and iteration to build a tower by using the agent. It shows a **for** loop. Loops allow you to repeat code. The **for** loop is special because you can easily monitor each pass through the loop.

Loops vs. functions

Functions, covered in Unit 6, also allow us to repeat code. Loops are different from functions.

- Loops repeat code over and over in one specific place
- Functions are used to spread repeating code throughout applications.

This activity about loops aims to help you understand how loops work and how you monitor each pass through a loop.

Import the tutorial to MakeCode from here:

https://minecraft.makecode.com/?ipc=1#tutorial:github:mojang/educationcontent/CodingWithMinecraft/unit_7/lesson_C

Challenges

Last step... Now we need to repeat this line four times. Can you figure out how to accomplish what's shown in the following picture? Well, besides the water part! You can make your square wherever you want.



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?

Hint: The unit 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?



Write your ideas here:

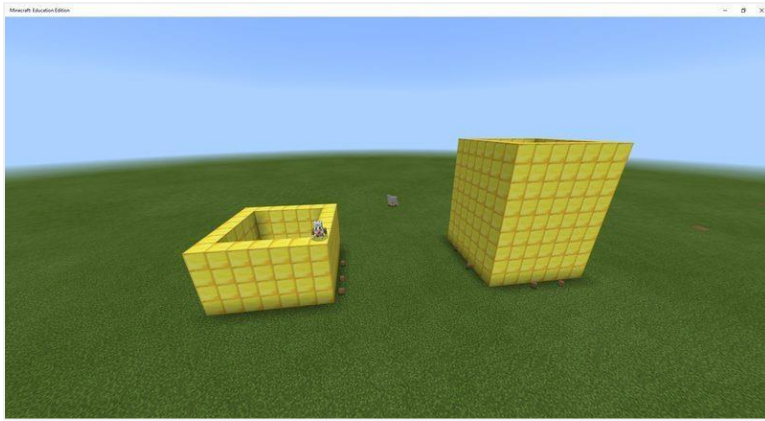
Extensions

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

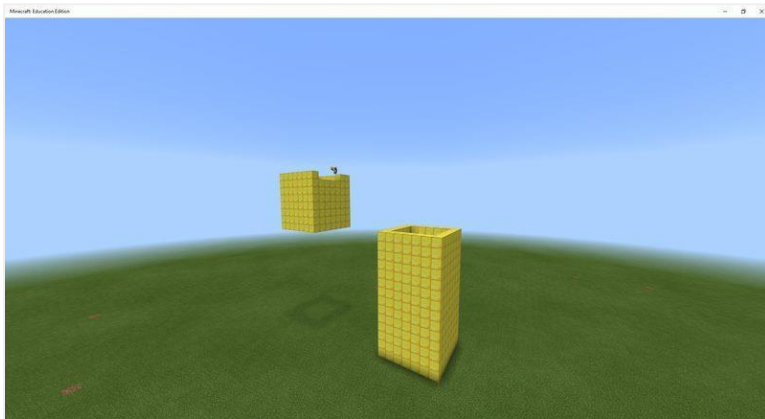
Extension 1 - Starter castle

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!



Write your ideas here:

LESSON C: DEBUGGING WITH ITERATION

Today's activities

- Debug a program
- Take a quiz

Coding activity: Help Your Agent Farm

You want to start farming with your robot agent, but there are a few errors in the instructions. Using loops, it would be nice to teach the agent to build a couple of rows of tilled soil where crops could be planted. Can you debug this code and figure out where the problems are?

You have your new garden all planned and need a 2 x 6 plot of perfectly aligned tilled soil. With a couple of `repeat` loops, you can get your agent to till this. But the code is only partially complete. You will need to go through the activity and fix the broken code at the end.

Import the tutorial to MakeCode from here:

https://minecraft.makecode.com/?ipc=1#tutorial:github:mojang/educationcontent/CodingWithMinecraft/unit_7/lesson_B/help_agent

Challenges

Challenge 1 - Get Agent to till aligned rows

There are a few ways to do this, but try to use the core of the existing code to do it. This means you need to keep the two loops in your answer.

Challenge 2 - 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.



Write your ideas here:

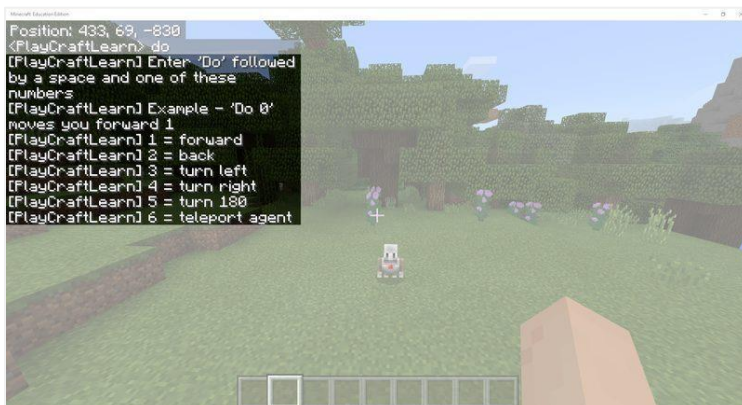
Extensions

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

Extension 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"**? For example, what if you wanted to enter something like **do 1 4** to move your agent forward four? What other code could you build a Help menu for?



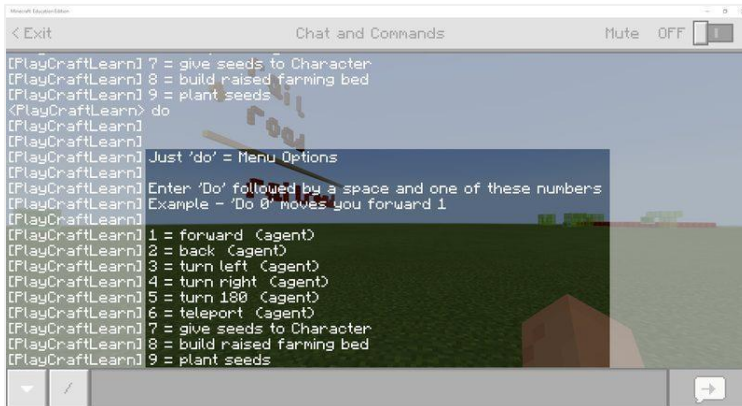
Extension 2 - Creating a Raised Farming Bed with Planted Crops

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. Functions are covered in unit 6, but basically, they are an awesome way to organize your thinking when you code.

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?

Write your ideas here:



LESSON D: GET CREATIVE WITH ITERATION

Today's activities

- Design your coding project
- Complete and turn in your Minecraft diary entry

Coding activity: Independent project

Remember, it's important to challenge yourself:

- Look at code you don't understand and try to take ideas from it.
- Copy and paste, change it, and make it your own.
- Think of outside ideas that you might try that might not be mentioned in lessons.
- Be able to take an abstract goal of just a few words and make code to accomplish it.

This project is an opportunity for you to show what you know. In this unit, you learned about how **LOOPS** can reduce the size of your code by repeating certain tasks. Now, your challenge is to design 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 layer 10-13 where diamonds are usually found is challenging. Luckily, you can program your agent to do this for you! 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. Mining with the agent is also covered in Unit 5: Conditionals.

Project expectations

Your project should do the following:

- Use loops in some way
- Code your agent to dig a tunnel down around layer 10-13
- Use one of the staircase options for the tunnel

In Minecraft, it's generally not a great idea to dig straight down, because there are caverns underground and even if your agent does it, there isn't always a place to put a ladder to climb back up. A stair-step approach is usually better, but you may still have to clean up after the agent in case it runs into a cavern where there aren't any blocks, or if it runs into granite or sand, which can fall and obscure the stairs.

Staircase options

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.

OPTION 1 - STRAIGHT STAIRCASE

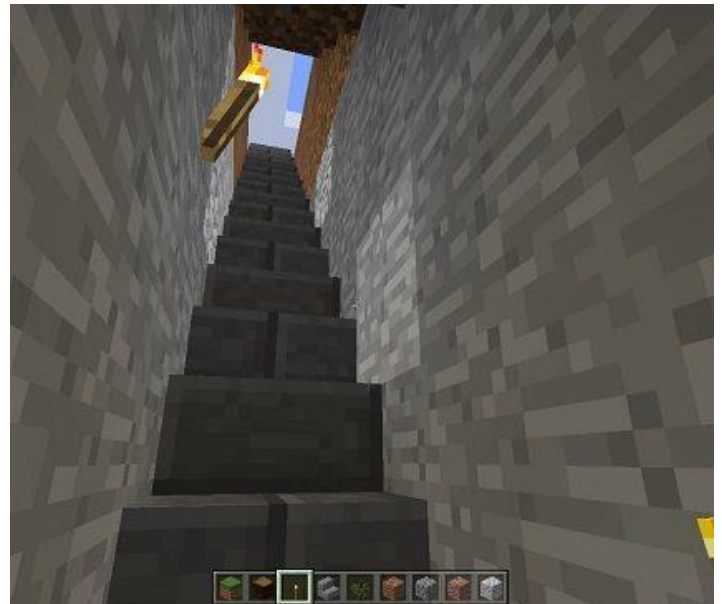
A straight staircase is most common, and it has the added benefit of allowing you to run up and down quickly if you line the rock with stairs.



Looking down the stairs:



Looking up the stairs:



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

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

Perhaps you can work around these issues.

If you do decide to put down stairs...

- 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.
- 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.

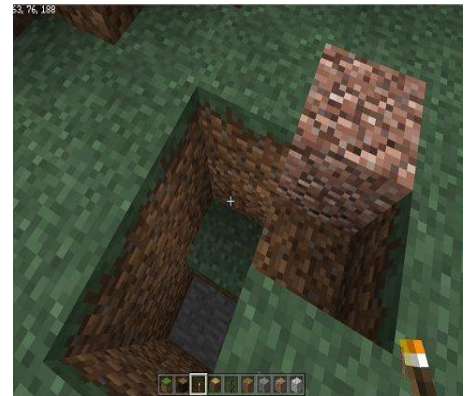
Remember to use **LOOPS** to reduce repetition of code!

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!

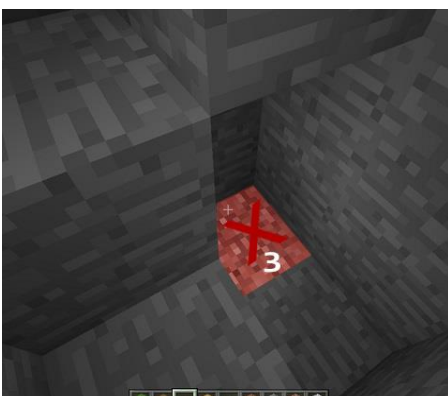
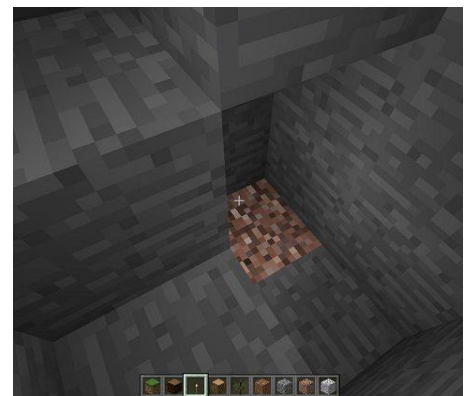
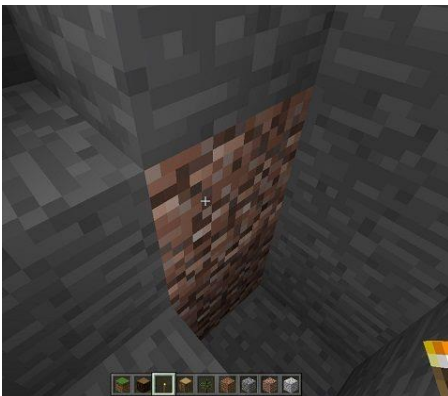
OPTION 2 - SPIRAL STAIRCASE

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 two blocks, and then three blocks deep:



After that, at each turn, remove a column of three blocks. Remove the block directly in front of the agent (block 2 in picture) and the blocks directly above and below that (blocks 1 and 3):



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

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

Remove the blocks indicated by granite:



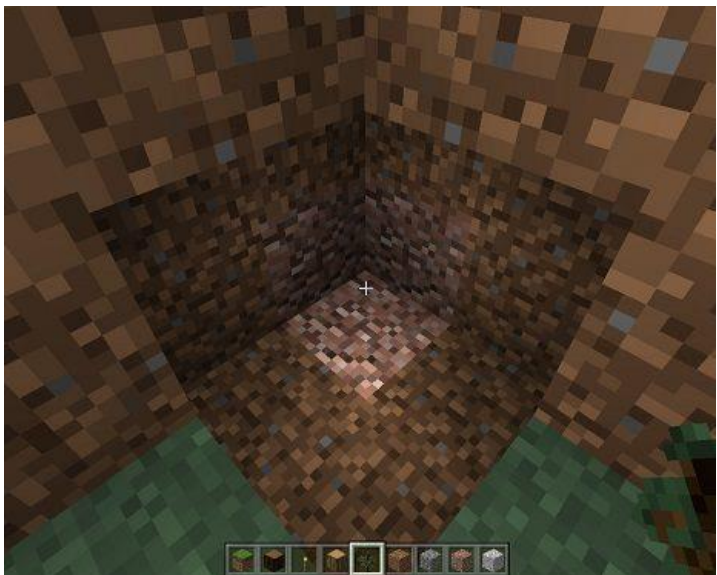
Remove the blocks indicated by diorite:



Remove the center block:



Continue digging a diagonal tunnel:



Tunnel is dug:



Remember, 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?

Minecraft diary

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

Assessment	1	2	3	4
Loops	Uses loops effectively in a way that is integral to the program.	Uses loops in a way that is integral to the program, but there are some problems with loop output.	Uses loops effectively but in a superficial way.	Doesn't use loops at all or uses loops in a superficial way and has problems with loop output
Project	Staircase is complete and navigable in both directions. Ends around layers 10-13.	Staircase lacks one of the required elements.	Staircase lacks two of the required elements.	Staircase lacks all of the required elements.

Assessment	1	2	3	4
Diary	Minecraft diary entry is missing four or more of the required prompts.	Minecraft diary entry is missing two or three of the required prompts.	Minecraft diary entry is missing one of the required prompts.	Minecraft diary addresses all prompts.

Write your ideas here:

GLOSSARY OF KEY TERMS

Iteration: Code that tells a computer to REPEAT sets of instructions under different conditions.