Introduction to Computer Science with MakeCode for Minecraft

Lesson 5: Iteration

In this lesson, 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 introduce you to 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.

What is Iteration?
Computer programming teachers often say, "Iteration is repetition. Iteration is repetition. Iteration is repetition." You get the point. To repeat is to iterate. To iterate is to repeat. And repeating is something that computers do well and very quickly.

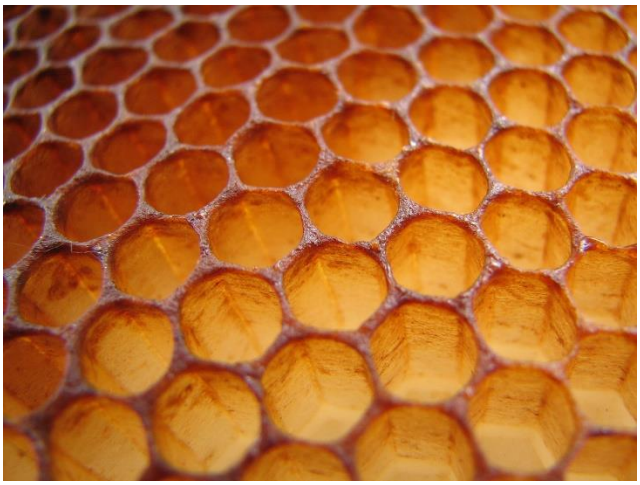
> Side bar: Iterate versus Reiterate
> Most people are more familiar with the word reiterate than the word iterate. Since iterate means to repeat, reiteration means to 're-repeat', a bit redundant, but solidly in place in common English speech. In programming, iterate and iteration are the terms used.

What are some real-life examples?
Example: Walking. We don't think much about it, but our bodies repeat the same sequence of actions whenever we walk. The basic visible action of walking breaks down to repeating the actions 'left foot forward', 'right foot forward'.

What are some other examples of iteration in your daily life? What tasks and actions are composed of repeated steps? What are those steps?

Architects use repetition in their building designs

Dogs love to play fetch as a repetitive activity



There are many iterative patterns found in nature

Unplugged activity: Everyday Tasks Part 1

Have students write down a task and then on a separate sheet of paper describe the repeated steps that comprise this task.

Have them swap their list of repeated steps with a partner. Each student should try to figure out what task is accomplished by following the repeated steps given to them by their partner.

Note:
Depending on the age of the students you are teaching and your time, you may wish to either:

1) Provide students with tasks, rather than have them come up with their own, especially since this helps avoid the same tasks being used more than once
2) Share the tasks written out as repeated steps as a class. You could have a student/or all the students act out the steps as you read them out loud

Some examples of tasks:

- Nodding your head yes
- Shaking your head no
- Waving to someone
- Walking up (or down) stairs
- Doing a specific exercise like jumping jacks
- Brushing your hair/brushing your teeth
- Eating soup from a bowl with a spoon
- Using scissors

Unplugged activity: Walk Around the House

This activity works best with a partner. You will be writing instructions for your partner to walk completely around a Minecraft house, and return to the start. You will need a chair, and some way to write stuff down. Place the chair in the center of the room. The chair will represent 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:

Forward ()
Turn left/right ()

A series of instructions that a computer can follow is called an *algorithm*. We can only process one instruction at a time, so the algorithm needs to be step by step.

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 go ahead and follow your algorithm to prove that it works.

But notice that this is eight lines of code, and many the lines repeat. In particular, the following two lines of code repeat four times:

forward()
turn left()

Programmers like to take shortcuts. Fewer lines of code take 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:

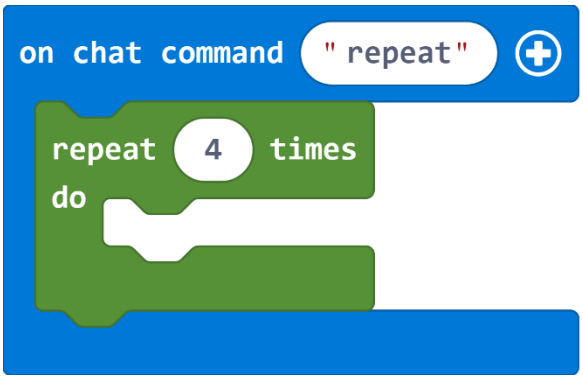
Repeat 4 times:
forward().
turn left()

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

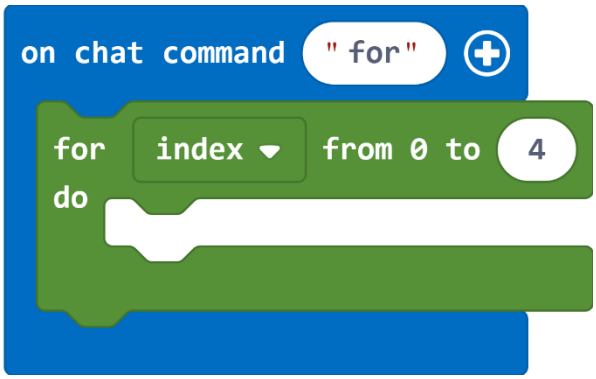
You have just rewritten eight lines of code as three lines of code, by using a loop. The 'repeat' command creates a loop. The code within the loop gets repeated a certain number of times until a condition is met. The condition in this algorithm is that the code in the loop is repeated four times. Once this condition is met, 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.



Repeat n times



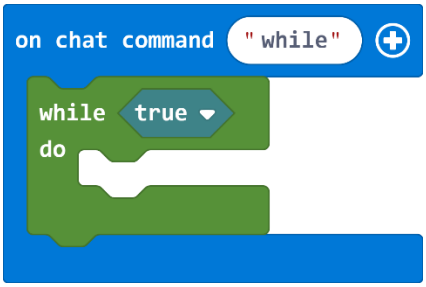
For index 0 to n

Both of these loops (Repeat and For) repeat n number of times. If you have 32 stairs in your house, then going upstairs or downstairs would probably involve taking 1 step 32 times. Other examples of activities that repeat a set number of times: Entering a building with two front doors, Lacing your shoes, 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 (by default; you can change it) that you can use inside of the loop if you need a number that changes each time through a loop.

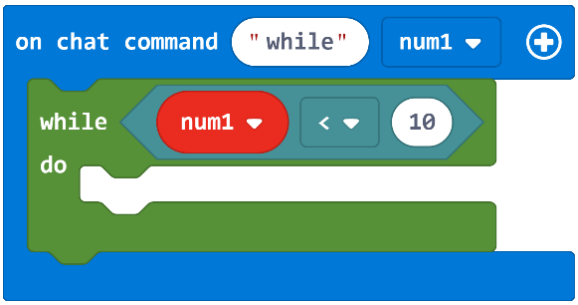
Fun fact: In JavaScript, which is the language underlying MakeCode's block-based language, the Repeat block is actually the same as the For loop! It just hides the loop variable to make things simpler. You can see this for yourself by clicking on the JavaScript button at the top of the window.

```
player.onChat("repeat", function () {
    for (let i = 0; i < 4; i++) {

    }
})
player.onChat("for", function () {
    for (let index = 0; index <= 4; index++) {

    }
})
```

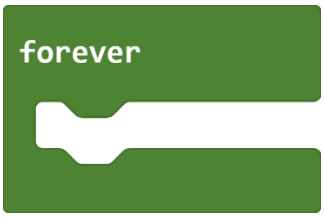


While <true>

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



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....



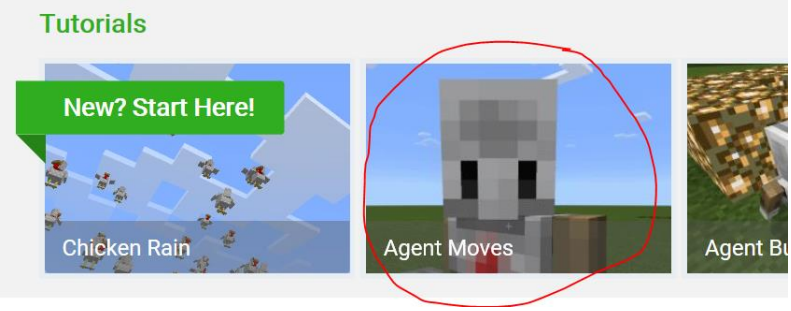
Forever

A Forever loop starts running when the program starts, and keeps going until the program ends. Some real-life examples are: Breathing; or your Heart beating.

Activity: Introduction to the Agent

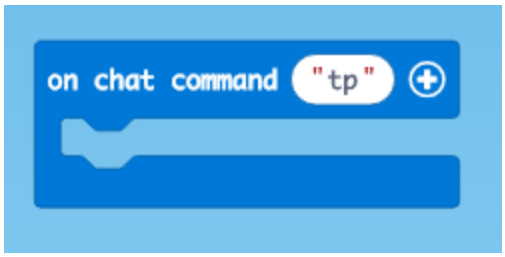
The Minecraft Agent is a little robot who can carry out the commands you write in MakeCode. This activity will walk you through getting started with the Agent.

Alternately, you can follow the interactive Agent Moves tutorial in MakeCode.

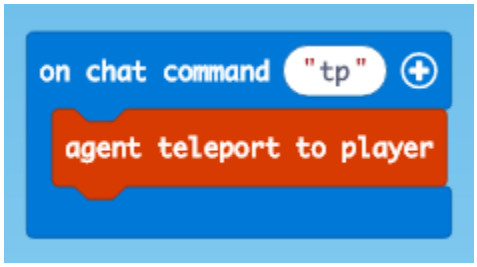


Steps:

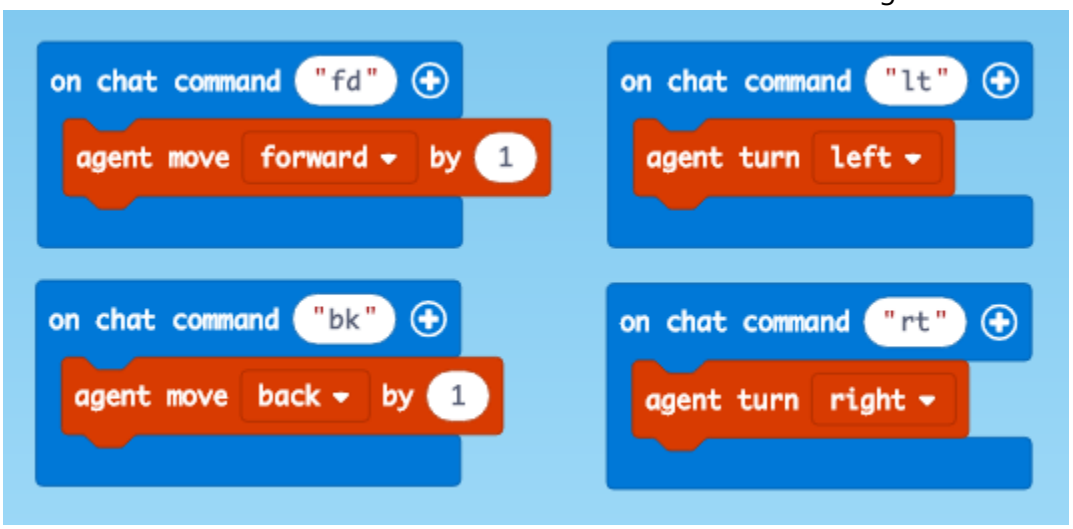
1. Create a new MakeCode project
2. Name the default On chat command block to "tp"



3. From the Agent Toolbox drawer, drag an Agent teleport to player block and drop it inside the On chat command block



Now, in Minecraft, when you type “tp” in the chat window, the Agent will teleport directly to your location. You should do this whenever you want to use the Agent in a project. You may also want to create some additional basic commands to move the Agent around:



This makes it easier to control the Agent precisely. Alternately, you could always just stand where you want the Agent to be and then type “tp” in the chat window.

Activity: Dance Dance Agent

In this activity, we’ll create a unique dance for the Agent to do, using Repeat loops and For loops. In the Agent Toolbox, you will see the following commands:

Move (forward) by 1

Turn (left)

You can select a different direction for each of these commands by choosing from the pull-down menu. With the Agent move command, the direction the Agent is facing doesn't change, so if for example, you want the Agent to slide side-to-side, you can use the Agent move command.

With the Agent turn command, the Agent will actually turn to face in that direction, so if for example, you want the Agent to turn in a clockwise circle, you would just tell the Agent to turn right four times in a row.

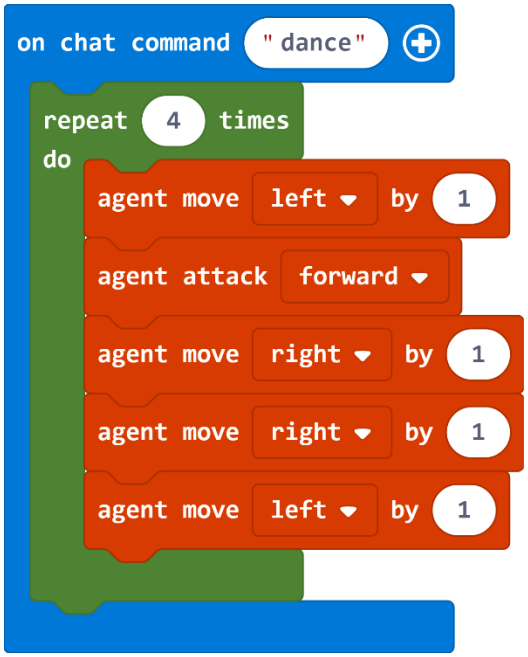
There are also these additional commands, which perform specific useful actions in other contexts, and make the Agent pump its fist or wave a tool:

For our dance purposes, feel free to mix some of those commands in to make your dance more interesting. Let's get started!

Steps:

1. First, from the Player Toolbox drawer, drag a new On chat command block to the coding Workspace
2. Name this command 'dance'
3. Then create a sequence of moves for your Agent to follow. Here is one example:

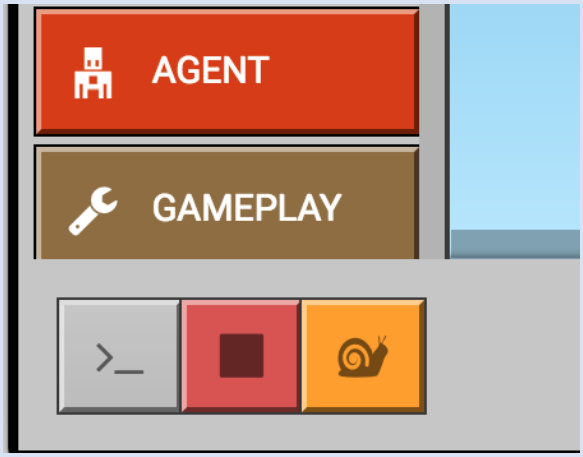
4. From the Loops Toolbox drawer, drag a Repeat block into the On chat command dance block and surround the Agent blocks



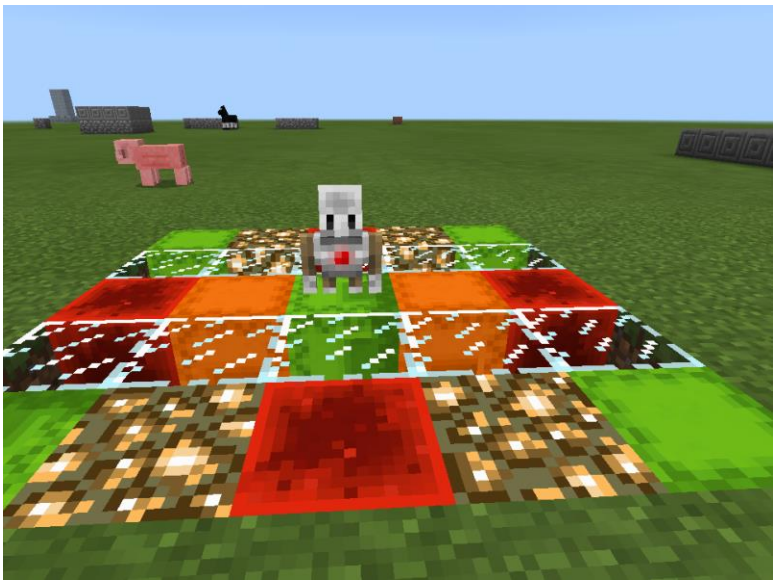
Use the Repeat block to repeat your sequence of dance moves a number of times. Then go into Minecraft, teleport the Agent to your location, and take a look at what your dance looks like!

Side bar: Slo-Mo
Because the Agent moves through the dance steps so quickly, sometimes it can be difficult to see what's going on if you are trying to troubleshoot a dance routine. MakeCode has a Slo-Mo feature that allows you to slow down a program, so that you can step by step as each instruction is carried out.

The Slo-Mo button is at the bottom left of the MakeCode screen, and has an icon of a snail on it. When you click to toggle it on, the button turns yellow. Now, when you run a MakeCode program, everything will run slowly enough for you to see each instruction highlighted in the coding Workspace as the robot carries it out in Minecraft.



Now, put all of your best dance moves together into a dance routine! How about creating a lighted dance floor to go with it?



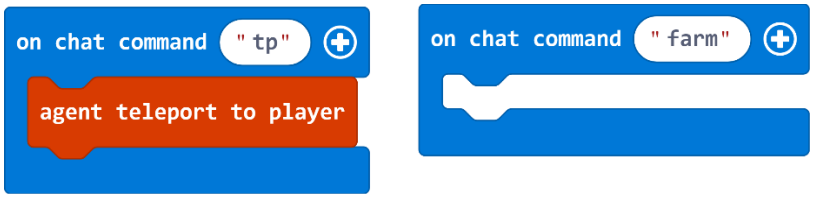
Activity: Help Agent Farmer

We're hoping to use iteration to teach the Agent to build a couple of rows of tilled soil where we can plant some crops. We'll need your help to debug this code (figure out where the problems are!) Let's get started.

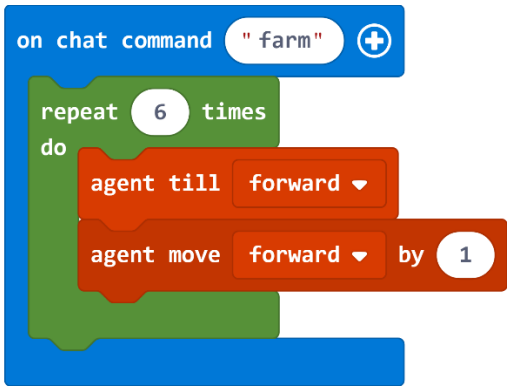
We'd like to create a 2 x 6 row of tilled soil. We can think of this as a couple repeat loops that gets the Agent to repeat tilling the soil and moving forward a total of 6 times, then have the Agent turn around and repeat those same steps again, for a second row of crops. You will have to figure out how to do that properly.

Steps:

1. Let's start out by creating a new MakeCode project called Farmer
2. From the Player Toolbox drawer, drag another On chat command block to the Workspace, so there are two On chat commands
3. Name one "tp" to teleport the Agent to your location, and name the other one "farm"
4. From the Agent Toolbox drawer, drag an Agent teleport to player block and drop it into the On chat command tp



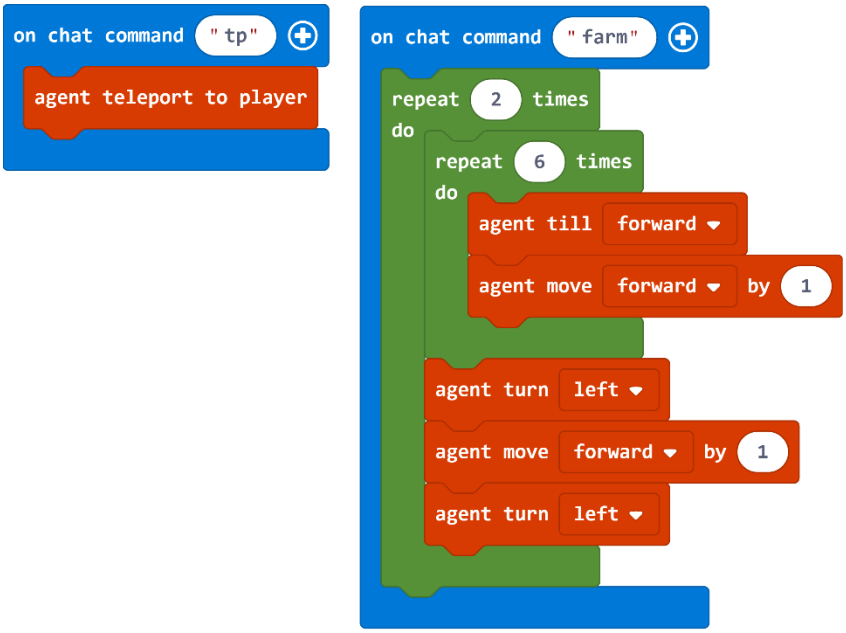
5. From the Loops Toolbox drawer, drag a Repeat block into the On chat command farm block
6. This will determine the length of our row, so in the Repeat block change the 4 to a 6
7. From the Agent Toolbox drawer, drag an Agent till block and an Agent move block into the Repeat block



This loop will repeat 6 times. Each time, the Agent will till the soil directly in front of it (with her own diamond hoe) and move forward into that space. Try running the code in Minecraft, and see what happens.

Notice where the Agent ends up after the Repeat block finishes. We want the Agent to turn around and till the next row over, until it ends up right next to where it started, and leave a nice 2 x 6 row of tilled soil ready for planting. You will need to add another Repeat block around your

existing Repeat block, and add some directions to turn the Agent around before it starts the next row. Here is our starter code:



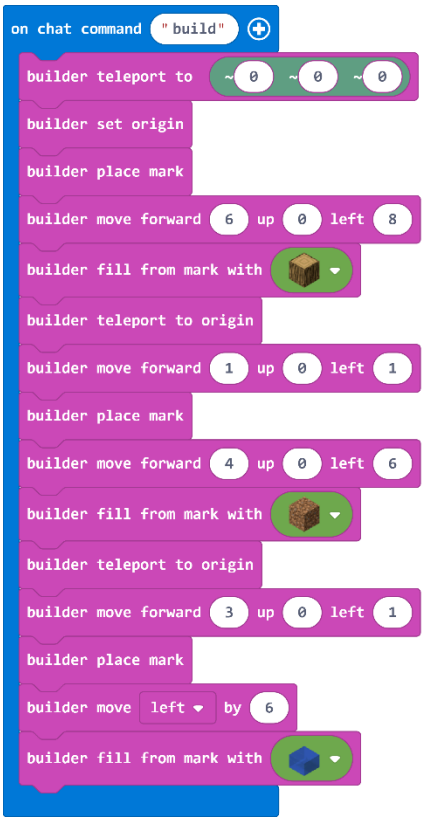
If you run this code, you will notice that there is a problem. The Agent is not set up properly and the second row is not aligned. How can you fix the code to make this work properly? We'll leave that to you to figure out!

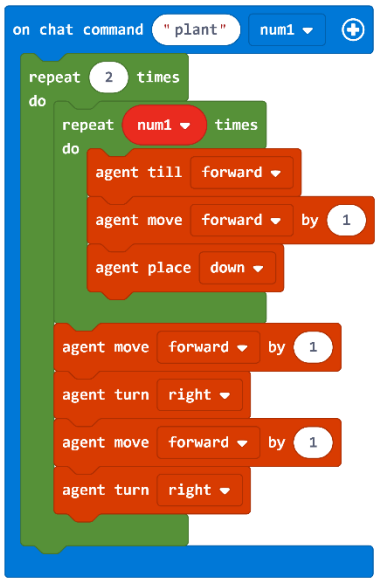
Teacher Solution: https://makecode.com/_2z3Hg6W4U1YV

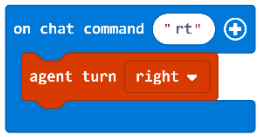
Optional Extensions

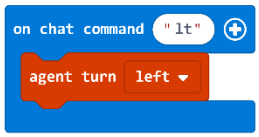
- Crops need water in order to grow. Find a way to create a 1 x 6 row of water in between each 2 x 6 row of soil.
- Teach the Agent to plant some seeds for you after it tills the soil. Remember to give the Agent some seeds in the upper left inventory slot.
- Create a way to duplicate the raised farming beds found in villages. This is not easy, so try using the Builder to do it. We've included one solution below, but there are others. Try it out for yourself first!

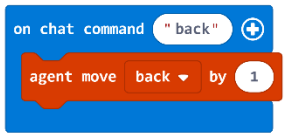
Solution to Raised Farmbed and Agent Planting

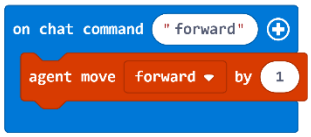












JavaScript:

```
player.onChat("build", function () {
    builder.teleportTo(positions.create(0, 0, 0))
    builder.setOrigin()
    builder.mark()
    builder.shift(6, 0, 8)
    builder.fill(blocks.block(Block.LogOak))
    builder.teleportToOrigin()
    builder.shift(1, 0, 1)
    builder.mark()
    builder.shift(4, 0, 6)
    builder.fill(blocks.block(Block.Dirt))
    builder.teleportToOrigin()
    builder.shift(3, 0, 1)
    builder.mark()
    builder.move(SixDirection.Left, 6)
```

```
    builder.fill(blocks.block(Block.Water))
})
player.onChat("plant", function (num1) {
    for (let i = 0; i < 2; i++) {
        for (let i = 0; i < num1; i++) {
            agent.till(SixDirection.Forward)
            agent.move(SixDirection.Forward, 1)
            agent.place(SixDirection.Down)
        }
        agent.move(SixDirection.Forward, 1)
        agent.turn(TurnDirection.Right)
        agent.move(SixDirection.Forward, 1)
        agent.turn(TurnDirection.Right)
    }
})
player.onChat("tp", function () {
    agent.teleportToPlayer()
})
player.onChat("rt", function () {
    agent.turn(TurnDirection.Right)
})
player.onChat("lt", function () {
    agent.turn(TurnDirection.Left)
})
player.onChat("back", function () {
    agent.move(SixDirection.Back, 1)
})
player.onChat("forward", function () {
    agent.move(SixDirection.Forward, 1)
})
```

Shared Program: https://makecode.com/_FTA0EuARH7Ys

Independent Project

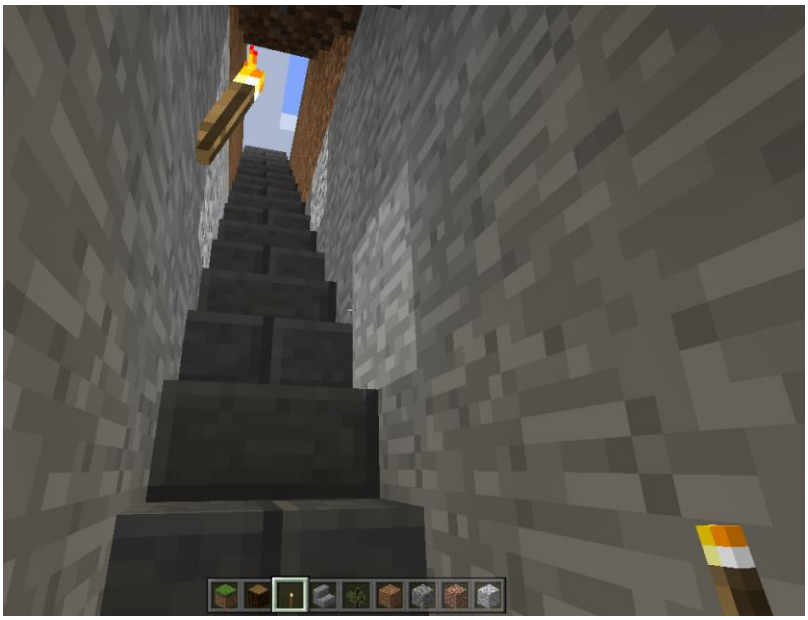
An independent project is an opportunity for you to show what you know. In this lesson, we learned about how the Loops blocks can reduce repetition in your code. Now, your challenge is to come up with a MakeCode project that uses iteration in some way to create a stairway to diamonds.

One of the most valuable resources in Minecraft is diamond. 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 level Y12 or Y13, where diamonds are found. Once you are at that level, you can dig parallel tunnels in search of diamonds, or even program the Agent to do it for you (we'll cover this in the section on Conditionals.)

Note: 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.

<u>Staircase Project Option #1</u>
Straight Staircase
A straight staircase is the most like what we are used to seeing, 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 stairs.

This challenge is a little easier if you aren't placing stairs, but 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.

If you do decide to put down stairs, note that 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.

Also, remember that when the Agent place on move block is set to True, the Agent will place a block every time it moves. You may need to set the Agent place on move to False to position the Agent, and then set it back to True again when you want it to start placing blocks again.

Remember to use Loops to reduce repetition of code!

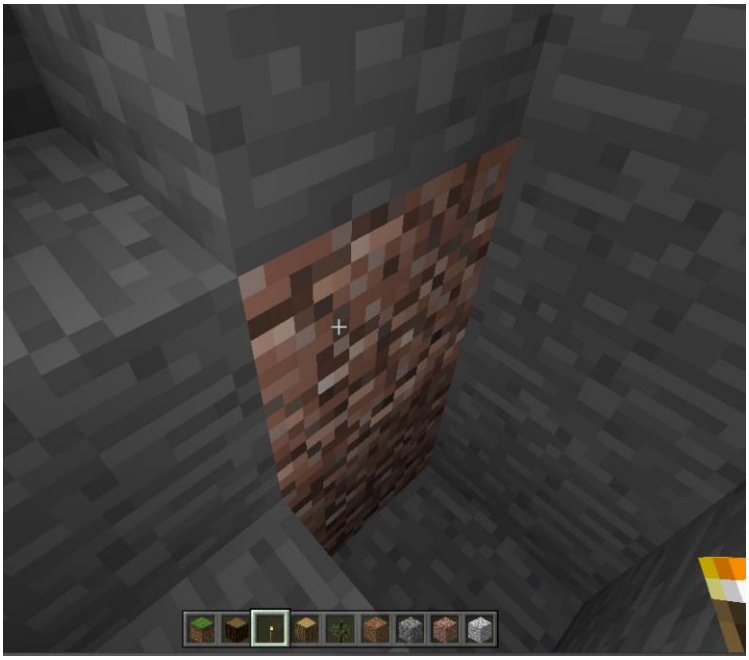
Once you have the Agent building a straight set of stairs, consider how you might modify your code to create a staircase three blocks wide and with enough headroom to allow you to ride a horse down the stairs!

Staircase Project Option #2

Circular 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, then three blocks deep.



At each turn, remove the column of three blocks indicated by granite in these pictures (the next picture shows the bottom block, not visible until the two on top of it are removed.)

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 after falling granite.

Staircase Project Option #3

Diagonal Tunnel

One effective type of tunnel goes down at a diagonal angle. Can you create some 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.

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.

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 here.

Assessment

	1	2	3	4
Diary	Minecraft Diary entry is missing 4 or more of the required prompts.	Minecraft Diary entry is missing 2 or 3 of the required prompts.	Minecraft Diary entry is missing 1 of the required prompts.	Minecraft Diary addresses all prompts.
Loops	Doesn't use loops at all or uses loops in a superficial way and has problems with loop output.	Uses loops effectively but in a superficial way.	Uses loops in a way that is integral to the program but some problems with loop output.	Uses loops effectively in a way that is integral to the program.
Project	Staircase lacks all of the required elements.	Staircase lacks 2 of the required elements.	Staircase lacks 1 of the required elements.	Staircase is: Complete Navigable in both directions Ends at or about Y13

CSTA K-12 Computer Science Standards

- CL.L2-05 Implement problem solutions using a programming language, including: looping behavior, conditional statements logic, expressions, variables, and functions.
- CL.L3A-03 Explain how sequence, selection, iteration, and recursion are building blocks of algorithms.