CODING WITH MINECRAFT 8: ARRAYS
Student workbook

education.minecraft.net

CONTENTS

OVERVIEW

Unit summary

In this unit, students will explore the concept of arrays. In computer science, an array is a series of places to store things. Earlier students learned how to use variables as a way to store information. Arrays store multiple values and they are accessed from a single object, the array itself. You'll continue with the idea of storing information for later use, but in this lesson, you'll use many values. The students will learn the usefulness of arrays as a collection of related items. You'll teach them how to use the array operations to add and get items to arrange and remember.

Learning goals

By the end of this unit, students will be able to:

- Explain the usefulness of arrays in coding as a collection of related items
- List examples of arrays in real life
- Use the array operations to add and arrange items
- Describe array indexes and elements in MakeCode
- Sort array values with basic sorting algorithms
- Design an original creative project that uses arrays to create a piece of artwork in Minecraft

LESSON A: INTRODUCTION TO ARRAYS

Today's activities

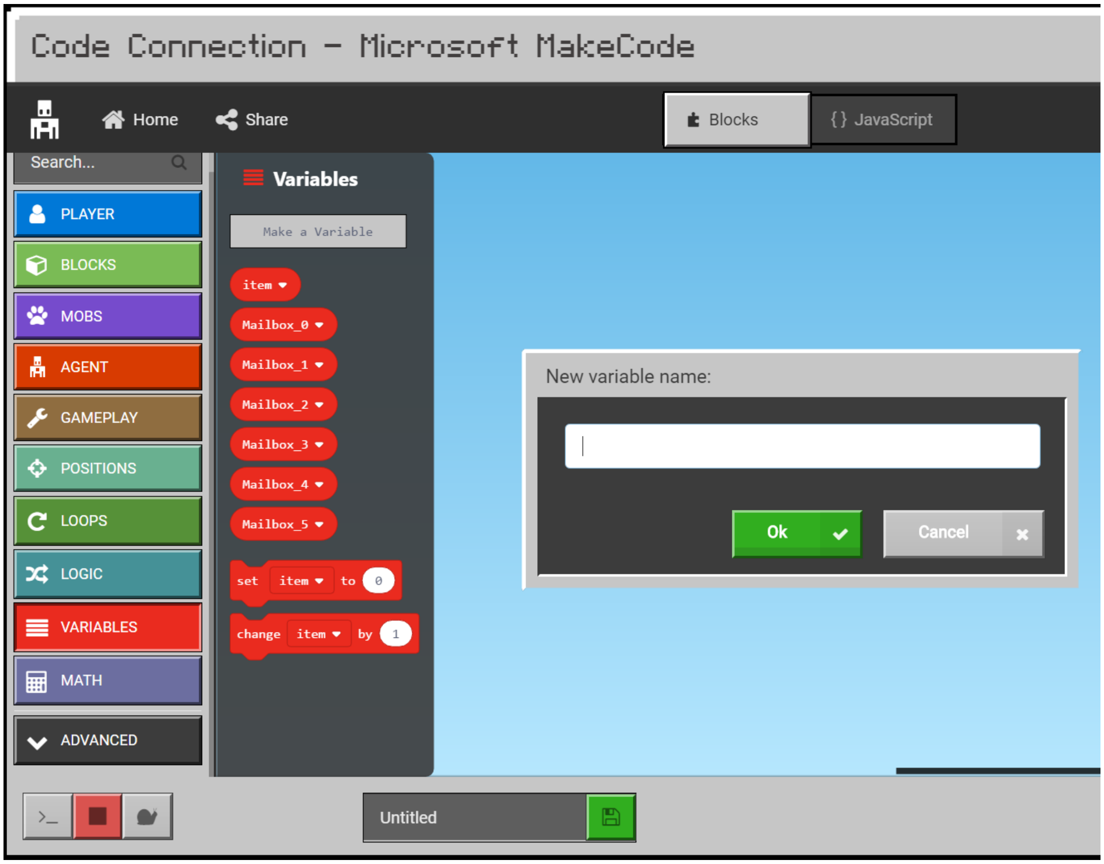
- Play two unplugged activities: Arrays in Real Life and Bubble Sort

Overview: Arrays

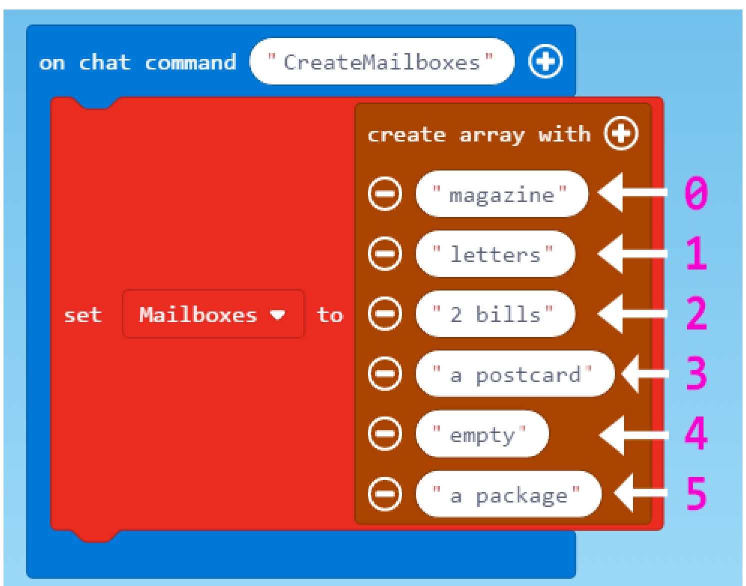
Previously, you discovered variables, which are used to store information. An array is a series of places to store things. You can think of it like many variables in one place. You can store several items under the same name and you only need that one name to find the information. This information could be a list of items, a row of mailboxes, or a train of container boxes. for example. The information in the array is all similar.

For example, maybe we want to store what each mailbox on a street has in it.

If you wanted to save information about the mailboxes and used a variable, you would need to make several variables with names like `Mailbox0`, `Mailbox1`, and so on.



This is time consuming because you have to type a lot, and it is inefficient when you want to later get that information back. This is where arrays help in a big way.



By using one array and one name, you can store all the information you need. Then you just use the one name, "Mailboxes," and the number or "key" of the mailbox you want. The keys in this example are `0`, `1`, `2`, `3`, `4`, and `5`. Sometimes the term index is used too. The key or index is the number of the box holding the information you want.

Therefore, when you have a lot of variables to deal with, it's more convenient to use an array. Also, arrays are handy when you don't know ahead of time how many variables you are going to need.

Array indexes

The location of a particular item in the array is known as its *index*. By default, arrays start at an index of `0`. The first item in an array has an index of `0`, the next one has an index of `1`, and so on.

The length of an array is the same as the number of items in the array.

For example, if there are six mailboxes, with indexes from `0` to `5`. The length of this array is `6`.

Array elements

In MakeCode, you can store different types in an array:

- Numbers
- Text
- Chickens
- Positions
- Blocks
- Zombies

You can add objects to an array, remove them, and count the number of things in the array at any given time. You can even reverse their order in the array with a single command. Arrays are a convenient and powerful way to work with lots of items at once.

Sort Array Values

After you start saving lots of different values in an array, you will probably want to have some way to sort those values. In computer science, there are certain common strategies, or algorithms, for sorting a collection of values. Understanding how those different sorting algorithms work is an important part of computer science. As you progress in your coding knowledge. you will learn about these algorithms, as well as their relative efficiency. Some of the different sorting algorithms include:

- Selection sort
- Insertion sort
- Quick sort
- Merge sort
- Bubble sort
- Shell sort

In this chapter, you will create an instant zoo, and fill it with animals from your array. You will also create a warp belt that allows you to save locations on your map and instantly teleport to them. Finally, you will use blocks in a new way with the power of arrays.

Write your ideas here:

LESSON B: BUILD A ZOO WITH ARRAYS

Today's activities

- Play a coding activity to experiment with arrays: We Built a Zoo

Coding activity: We Built a Zoo

You can store animals in an array and spawn them wherever you like. We'll use this capability to build a fenced-in animal pen and create an instant zoo anytime we want. When this project starts up, it will create an array and fill it with animals of your choice.

Import the tutorial to MakeCode from here:
https://minecraft.makecode.com/?ipc=1#tutorial:github:mojang/educationcontent/CodingWithMinecraft/unit_8/lesson_B

Challenges

CHALLENGE 1 - ADD MORE ANIMALS TO THE ZOO

Create a larger pen and add all the different types of animals! There are 19 different kinds. Then, watch the craziness ensue!

CHALLENGE 2 - CREATE A SECOND ARRAY FOR ANIMAL NAMES

This challenge applies several concepts in one. Are you up for it? Remember that learning is not always about winning. If you struggle with this challenge then don't feel discouraged. If you find this challenge too easy then perhaps you could make your own challenge and share it with another student!?

An ocelot is a very obscure animal to many so it would be nice to know the names of the animals we are spawning.

Create a second array that lists the names of the animals from animalarray. Then as the animals spawn use say to give feedback. Including a pause between animals will give the user more time to read and understand what the different animals are.

Tasks:

1. Create an animalnames array.

2. Put all the animal names into animalarray.
3. Print out the names after they spawn.
4. Pause the code after printing for 1 second or more… your choice on how long to pause.

Because there are so many animals to list for task #2, you may consider doing some of this in JavaScript instead of using blocks. After you create an array you can enter the names of the animals much easier in JavaScript. Plus you can see all the names much easier.

Block coding is awesome but, in this case, typing in JavaScript would be much faster. Look at the example below. Can you figure out how to copy this format for all the animals? Then when you finish you can add your say and pause by going back into the block editor.

Make sure your commas and quotation marks are all in the right place or you will get an error.

Tasks 1-2

The blocks below are the answers and might help you get the job done faster. Now you just need to figure out where to place them! In order to print the name of the animals you need to use a counter that acts as the key to each name. Remember the key is the number where things are put in the array…. for our new array, the keys will look like:

Chicken = 0
Cow = 1
Pig = 2
etc…

You will need to set the names for each animal in our new array. One example is given below. Now you just need to figure out where to place this code and finish setting the names for all the animals.

Tasks 3-4

Below you have been given the counter 'CurrentAnimalNum'. This acts as the key when getting values from the animalnames array.

If you don't understand all of this, it is not the end of the world. Just see if you can place this code in the right place. As you do this, try and understand why things are happening the way they are. If you need help on placement, ask your instructor for guidance.

Where would you place these blocks in our existing code?

HELPER BLOCKS

```
set animalnames ▾ to array of "Dog" "Cat" ⊖ ⊕
say animalnames ▾ get value at currentAnimalNum ▾
change currentAnimalNum ▾ by 1
pause (ms) 1000
```

Experiments

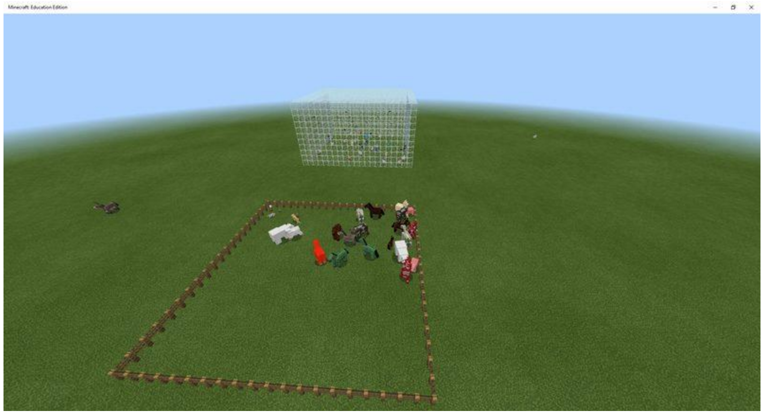
Let's add to our existing zoo to continue to make it bigger and better. Play around with this code then see what else you can add to make the zoo more complete!

EXPERIMENT 1 - ADD A BIRDHOUSE, BIRDS ONLY!

In this experiment, we separate the birds into their own array and build a cage for them specifically. An aviary is a building where birds are kept. This is what they will get. Type 'zoo' into the chat window and now you will get your land animals in their pen and birds in their own aviary. However, there are still some problems.

How might you make this better? Can you build a separate pen for each animal? What else can you think up to do?

Here you can see one problem. Look how the donkey is red. This is because our wolves are attacking them. Perhaps wolves need to be separated into their own pen or cave or something.

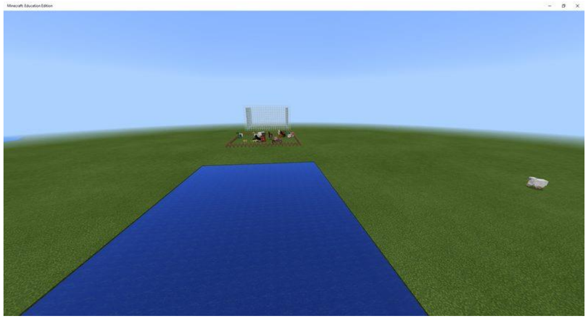


EXPERIMENT 2 - ADD ANOTHER ZOO EXHIBIT FOR THE SQUIDS

Since squids swim in water, this script builds a place for them to swim in the ground and then fills the ground with the water. Type 'zoo' in the chat window to get started.

Again, you might try to extend the zoo even more by adding signs, different features, caves, rivers, or any other thing you can think of.

Notice the code uses functions to help organize the new enclosures for different kinds of animals. Each function creates the pen and animals for each type of animal.



Write your ideas here:

LESSON C: TELEPORT WITH ARRAYS

Today's activities

- Play a coding activity to experiment with arrays: Warp Belt

Coding activity: Warp Belt

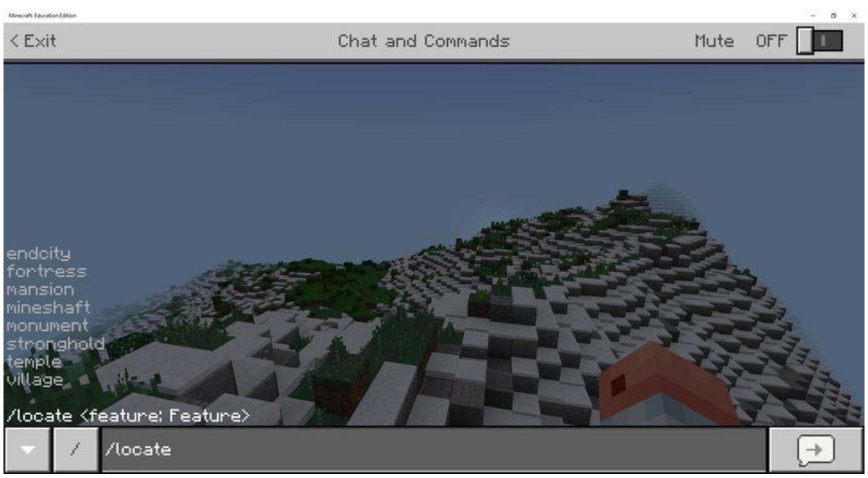
One of the most enjoyable activities when playing Minecraft is to build houses. Often, these houses are spread all over the world. Sometimes while exploring, you might discover awesome hidden temples or villages in remote locations. It can be challenging to remember how to get back to all of those places. In this activity, you'll see how to create a tool that you can use to teleport between all of those different locations. This "Warp Belt" will save the locations in memory by using arrays so you can easily jump between them all.

In the chat window, enter "/locate" followed by a space.

You can then enter in any structure you might want to find in the world you are in. You might use this to first locate things, teleport there, and then save the locations in your warp belt!

For example, to find a mansion in a world, enter the following:

```
/locate mansion
```



You'll support four main commands in this program:

- Delete: This creates an empty array, effectively deleting your old one.
- Save: This saves your current position to the next empty spot in the array.
- Warp: This command, when entered with a number, teleports the player to the position stored at that index in the array.
- List: This command prints all the positions in the array, with their index numbers.

Import the tutorial to MakeCode from here:
https://minecraft.makecode.com/?ipc=1#tutorial:github:mojang/educationcontent/CodingWithMinecraft/unit_8/lesson_C

Challenges

Let's continue to improve our warp belt code by checking what the user enters and giving them feedback.

CHALLENGE 1 - MAKE SURE A VALID JUMP SPOT WAS ENTERED

There is one more improvement you can do to make your program more user-friendly. If the player enters a number that is larger than the last index in the array, there should be some sort of a warning. Otherwise, it's odd to enter a command and have nothing happen. It is good programming practice to give feedback to the user for bad input or other errors.

You will need to do the following:

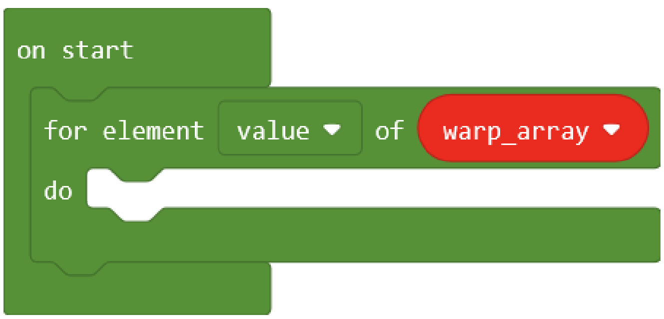
1. Add an if statement to check what was entered.
2. Check to see if value is bigger than size of `warp_array`.
3. If bigger, give a warning message like "Sorry, no such location".
4. If the same or smaller than size of `warp_array`, you will teleport the player.

You will need to set up your if statement and find a way to get the size or length of `warp_array`.

CHALLENGE 2 - PRINT ALL SAVED WARP COORDINATES

You need to print the list of all saved warp coordinates in a world.

To do this, you can use a special loop.



There are several ways to do this. You could make a function to handle this and call the function after each save. You could use an on chat command.

The basics are to loop through warp_array and print its contents. Again, with printing you have options. You could use say blocks to display the contents in the chat window, or you might write the results in the sky with blocks by using the print block.

The solution given uses a function. The function is called after every save.

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

Enter `save` in your chat window, and then press enter. You should see directions prompting you to enter a number now. You will now save coordinates and keep a name of the type of location connected to this.

Enter `list` to list all the locations you have saved.

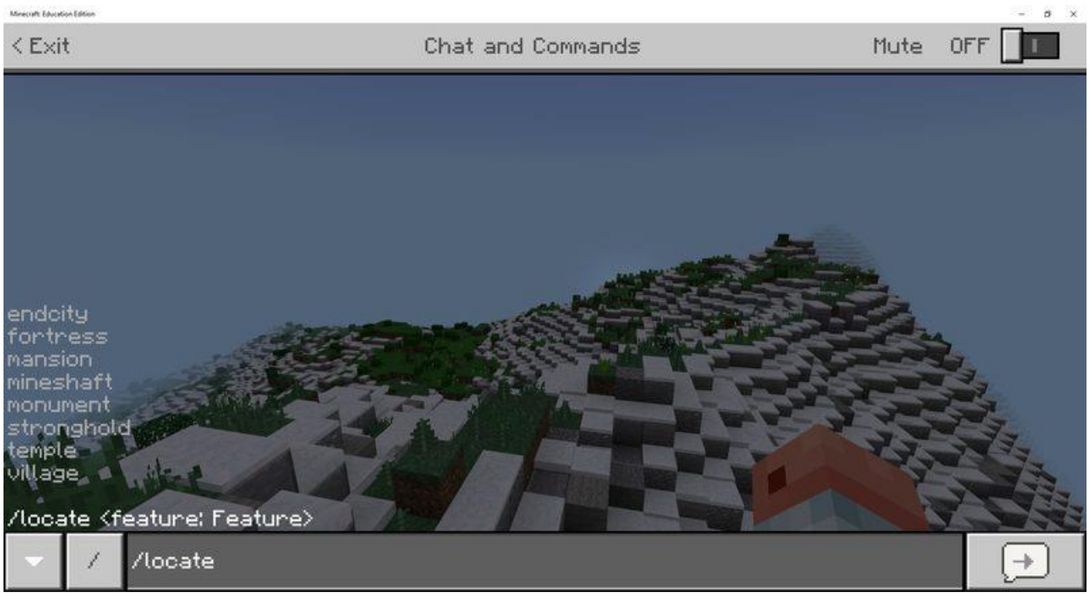
This code does multiple things. It uses multiple arrays that have connected information. This means the same index works with multiple arrays and the information you get is connected. In one array, the

In the chat window, enter '/locate' followed by a space.

You can then enter in any structure you might want to find in the world you are in. You might use this to first locate things, teleport there, and then save the locations in your warp belt!

For example, to find a mansion in a world, enter:

```
/locate mansion
```



coordinates are stored; in another, the names of the locations. A third array is used to keep the default locations, but it is not necessary to do it this way.

What places does it save?

What other things could you do with teleporting or coordinates in code?

Can you use arrays creatively to make the code more complex?

What about saving a monster type to each location? Then when you warp back, it spawns that monster! You could make a teleporting game with missions to teleport to different locations, kill all the monsters there (which you spawn with code), and then teleport to the next location.

Write your ideas here:

LESSON D: BLOCKS WITH ARRAYS

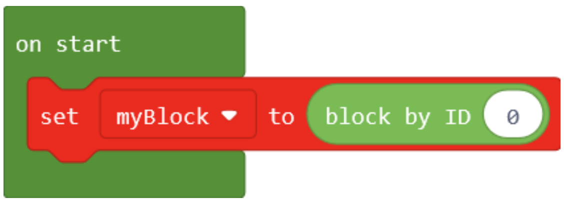
Today's activities

- Play a coding activity to experiment with arrays: Blocks by ID
- Take an Arrays Unit quiz

Coding activity: Blocks by ID

You can store an assortment of blocks in an array and access them by ID however you like. The ID is a number Minecraft uses to keep track of all the items, blocks, and other things in the game. We'll use the IDs for blocks and the power of arrays to store the information for us.

This project also features another block we have not tried up until this point.

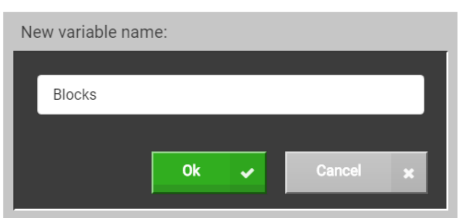


This allows us to supply a number, the block ID, and it will return the block for us. You can experiment with different IDs but be aware that our own experiments led sometimes to fatal crashes. Even finding a list of IDs on the internet sometimes yielded the wrong IDs but this is what experimentation is about. Still, tread with a bit of caution.

Do the activity

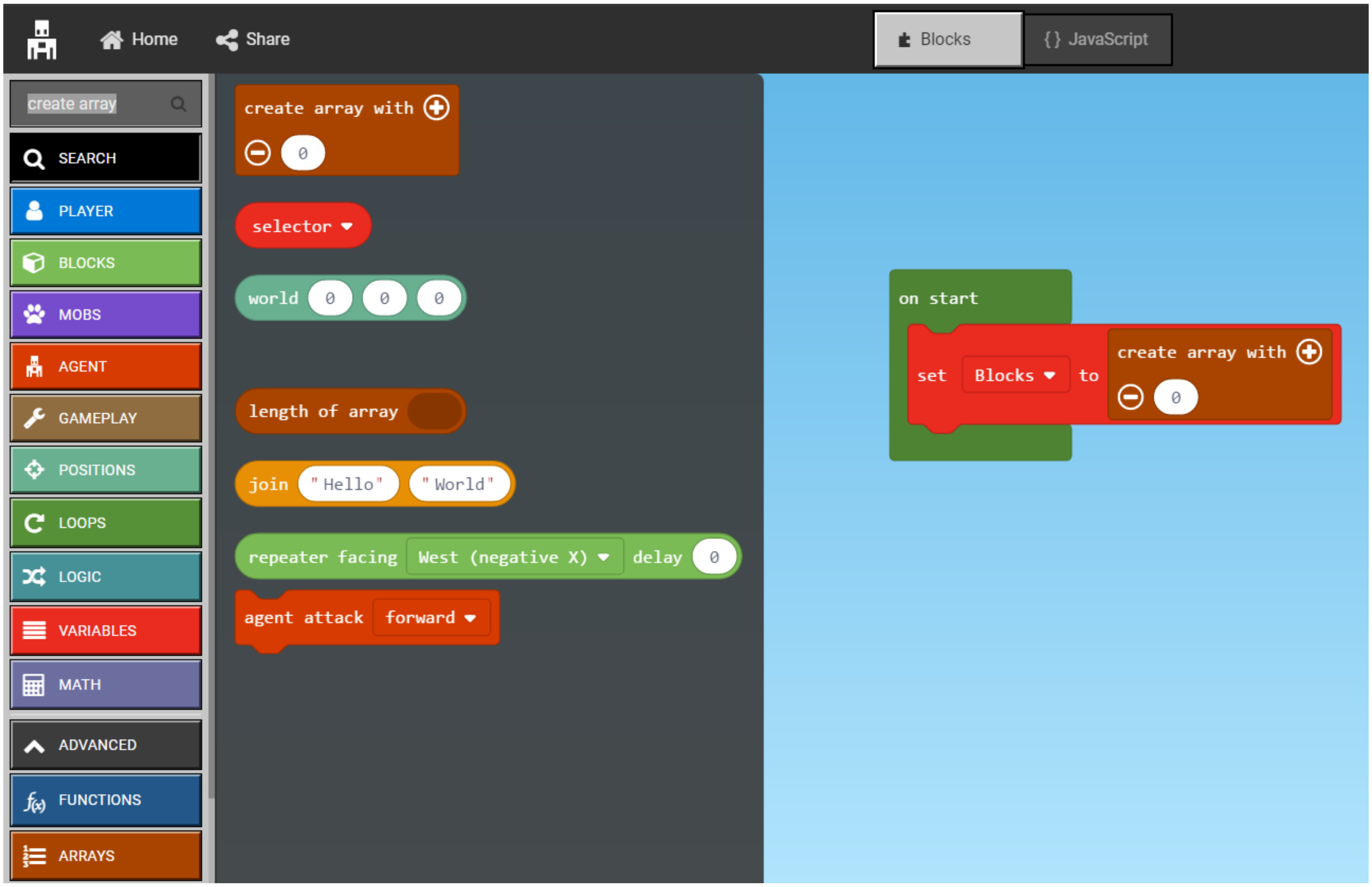
CREATE THE PROJECT

1. Create a new MakeCode project called "BlocksByID".
2. In the LOOPS Drawer, there is an on start block that will run its commands once, as soon as the project starts up. Drag that block into the coding Workspace.
3. From VARIABLES, select the **Make a Variable** button.
4. Name this variable `Blocks`, and select **Ok**.



MAKE AN ARRAY FOR BLOCK IDS

1. From VARIABLES, drag set into the on start. Adjust the variable from **item** to **Blocks**.
2. Using the Search, type `Create Array`.
3. Drag a create array with into set.

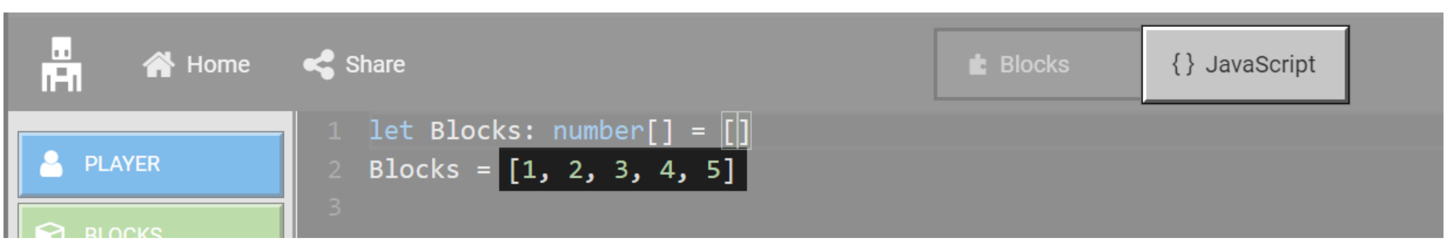


ADD BLOCKS TO THE ARRAY

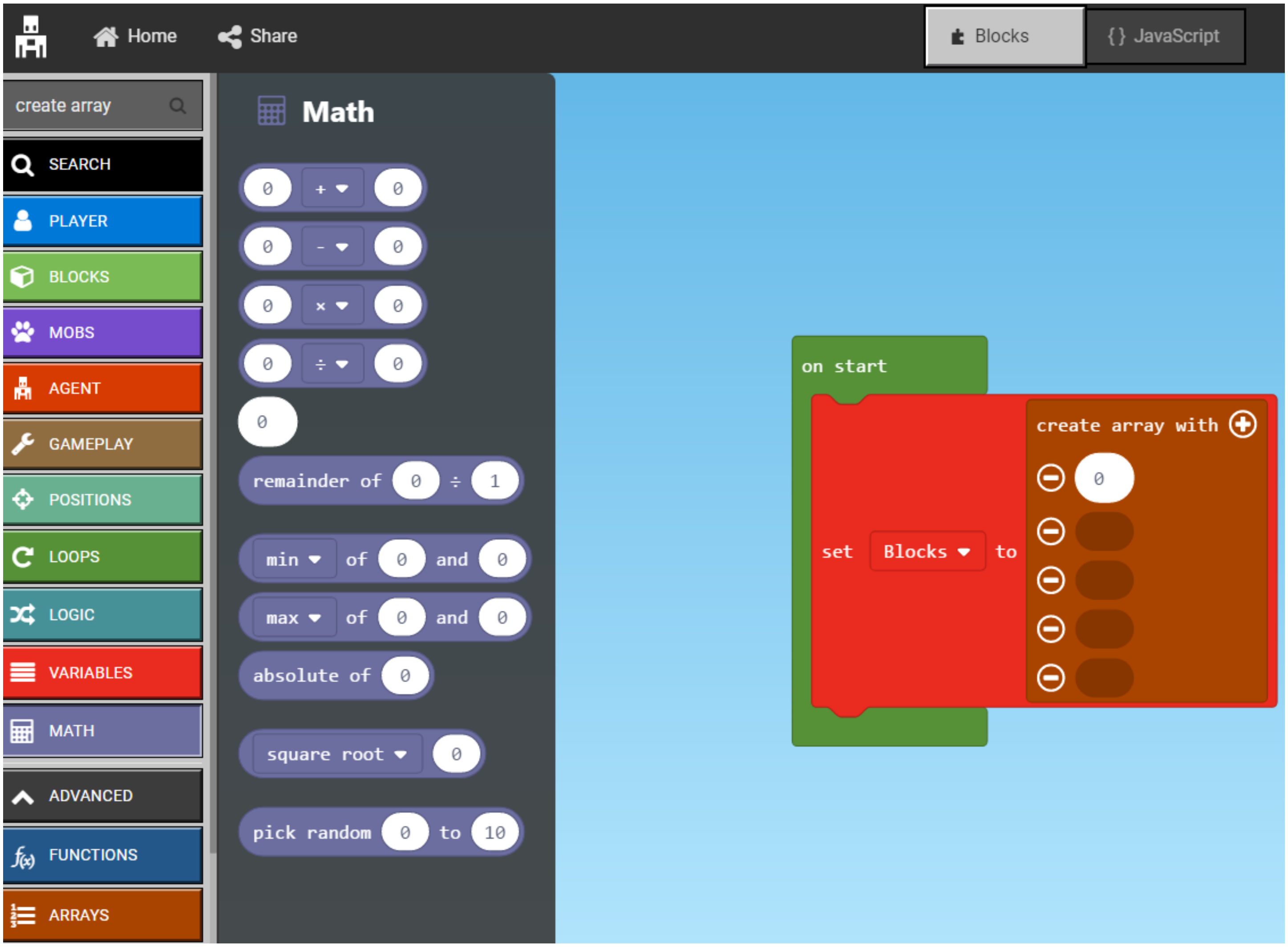
If you are feeling adventurous go to the JavaScript view and add five places in the array. It is a bit easier this way once you get past the code view! Each spot in the array will eventually be 1-5 as you will see later. Add the numbers 1-5 separated by commas. These five block IDs are:

1 = stone, 2 = grass, 3 = dirt, 4 = cobblestone, and 5 = oak wood plank.

If you do use JavaScript, you can skip steps 8-11.



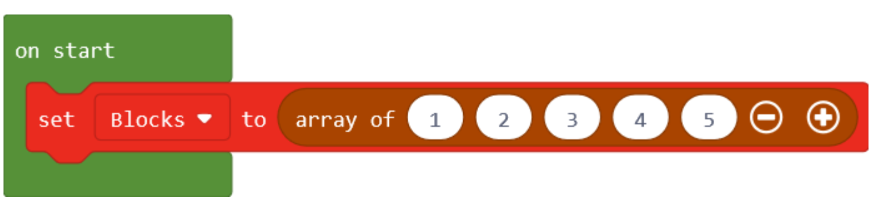
1. Select the Plus (+) sign on create array with to add 4 more slots in your array so the total length of your array is 5.
2. From MATH, drag a number block into the first slot of the create array with block.
3. Populate the rest of your array with number blocks. Note – you can right-click on the number block and select Duplicate to make copies of this block, instead of dragging more blocks from the Toolbox.



PICK YOUR NUMBERS

Create an assortment of 5 different types of blocks. Be aware that certain blocks will not mix well! For example, fire will burn all adjacent flammable blocks. This might be fun to play with or perhaps you want to make a design. The final project is about making art and this activity could be helpful with that so try out some different ideas.

1. Type the numbers to cycle through the first five block IDs: 1 = stone, 2 = grass, 3 = dirt, 4 = cobblestone, and 5 = oak wood plank. Again, you can search for a list of block IDs for the education edition or you could just try out different numbers from (1-252). For the most part you should be ok but again, do not be totally surprised if a fatal error occurs. Just restart.

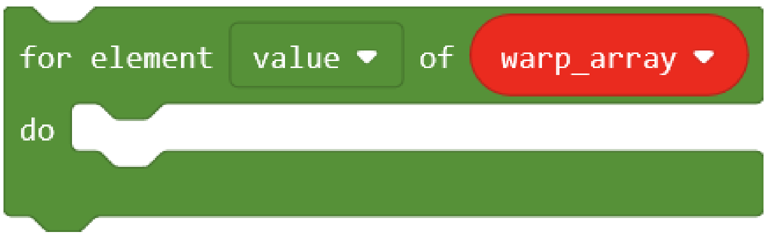


A LOOP TO PRINT OUT THE BLOCKS

Now that we have our Block Array set up, let's work on displaying the blocks in a for loop.

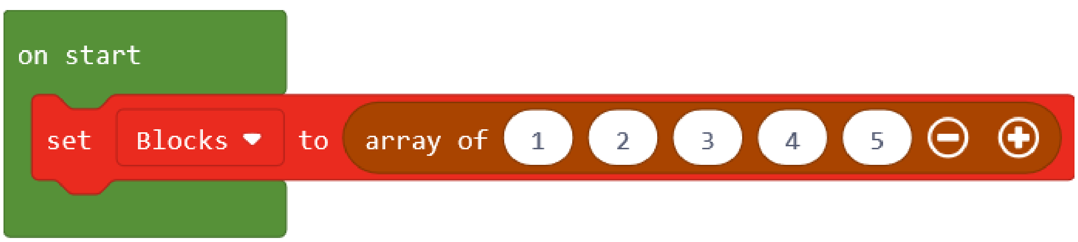
1. From PLAYER, drag an on chat command block to the Workspace.
2. Rename the command "PrintBlocks".
3. From LOOPS, drag a for loop and place in on chat command "PrintBlocks".
4. From MATH, drag a subtraction block out to replace the default number 4. We will subtract 1 from the length of the array.

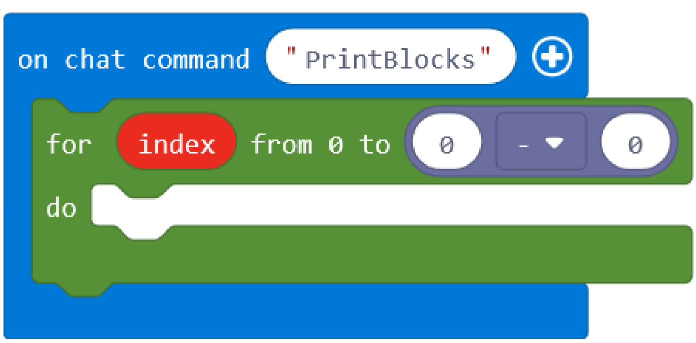
The reason you are using a for loop here is to see another way you might cycle through your loops and get the information. Recall in previous activities this loop was used:



This other LOOP is great! In fact, you might rewrite this code and try to use the for element of to do the same thing we are accomplishing here in this activity. This could be a personal challenge. However, you can learn more and try out more of the array blocks if we try different ways of accomplishing our goal.

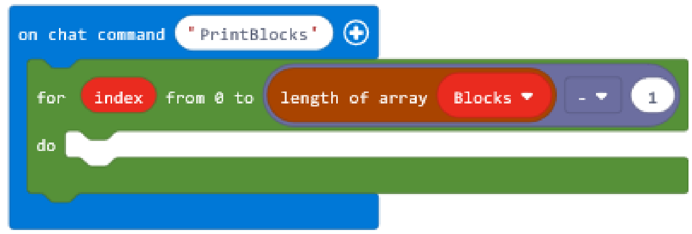
Your code should now look something like:





1. Search for length of array. This returns the number of values in an array. Place this in the first slot of your subtraction.
2. Add Blocks into length of array.
3. Add a 1 into the second slot of your subtraction.

Your code should look similar to this:

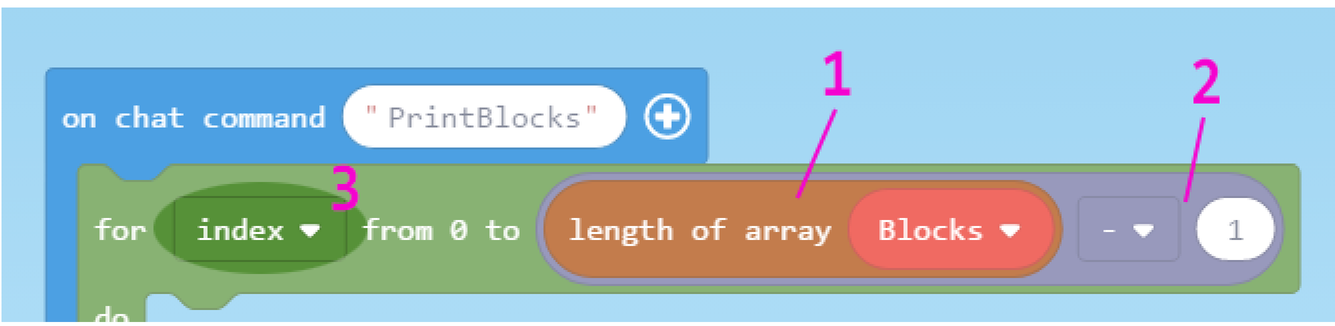


What is this doing?

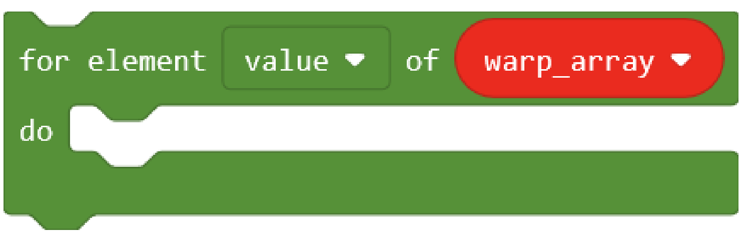
When examining long pieces of code, it is important to cut things down into smaller chunks. If it helps use paper and pencil to keep track of the logic.

Here we can see:

1. We are getting the length of our array which will be 5.
2. We subtract 1. This means the loop will now go from 0 to 4, (5-1).
3. Index becomes important. This is what we will use as the index or key for our array because now the numbers will align perfectly.



It is a bit more work than this loop:



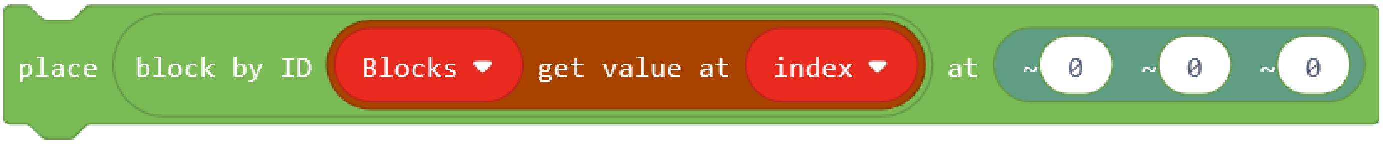
But by doing it this way, you could have multiple arrays that use the same key. In one of the experiments below the code does just that!

PRINT BLOCKS BY IDS

1. From BLOCKS, drag a place into the for loop.
2. Again, from BLOCKS, drag a block by ID out to replace the default grass block in place.
3. From ARRAYS, drag a get value at and place it inside block by ID.

get value at will look in our Blocks array and get our IDs(1-5) according to the key we feed it. So we have to adjust this just a bit more to get what we want. However, if you supply 0 for example we want it to give us the ID of 1.

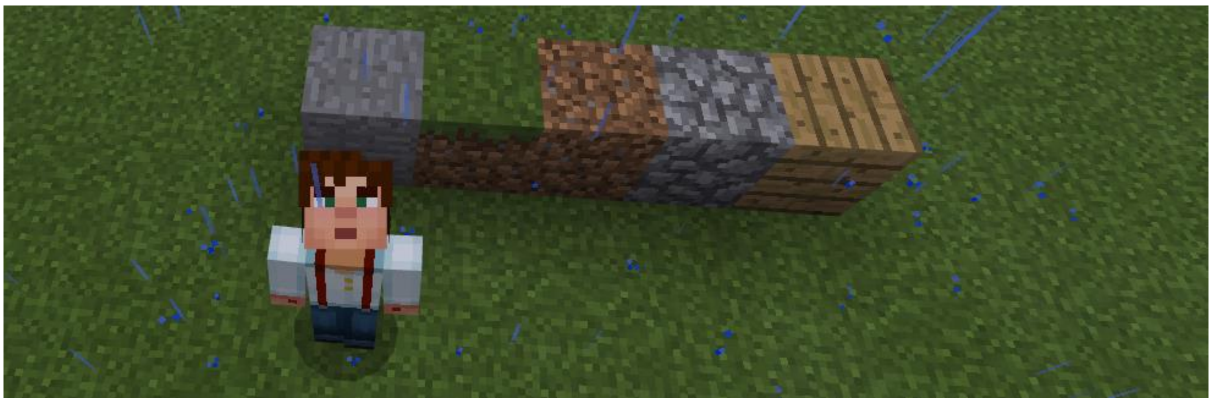
1. Adjust get value at so it reads "Blocks" get value at "index". You can get the variable Index from the VARIABLES drawer.



In the position parameters for place, set the XYZ coordinates as (~2, ~0, ~index). Drag Index and place it as the Z coordinate.

Try the "*PrintBlocks*" command

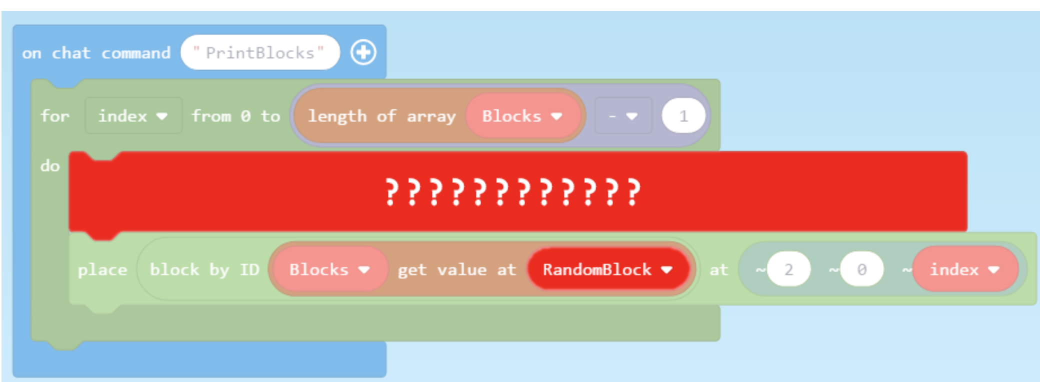
Open up a Flat World in the Minecraft game, and type "PrintBlocks" in the chat window. You should see the blocks being built! You can add to the array to print more blocks or change the numbers from 1-5 to any numbers between 1 and 252. Again, this is an estimation and things do change so be aware that if something goes terribly wrong, you should just not use that ID anymore!!!



Challenges

CHALLENGE 1 - RANDOMLY PRINT ONE OF THE STORED BLOCKS

Can you randomly print a block out of the array? You would need to build a variable called `RandomBlock`.



In that variable you would select a number randomly from 0 to the length of the array - 1. Remember we subtract 1 because arrays are zero based. There are other ways to work around this too if you want to try.

The solution posted here picks a random block from the array. It prints 5 blocks but you could print more if you wanted, right? How?

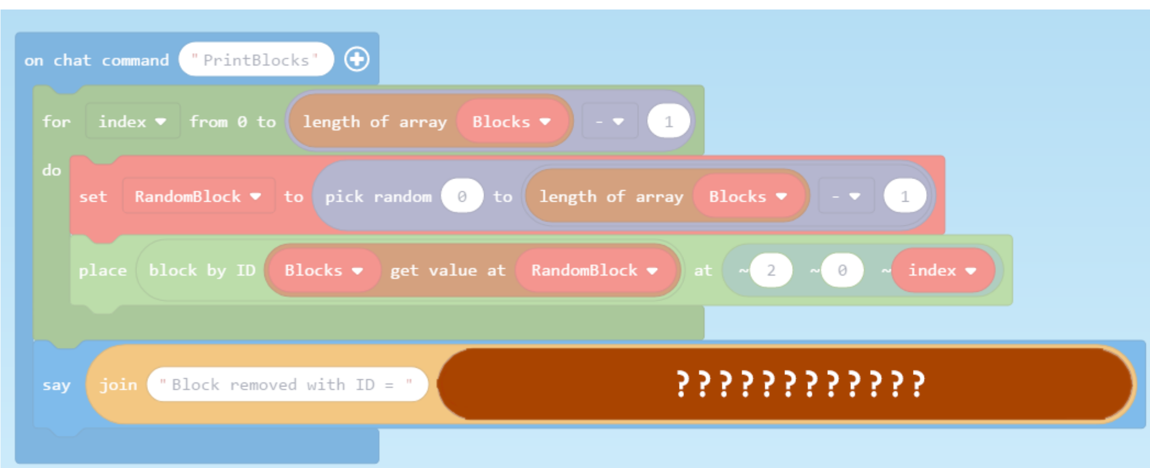
Some blocks might be printed twice because they are chosen randomly.

CHALLENGE 2 - REMOVE A BLOCK AND REPORT FEEDBACK

Thus far we have only added things to arrays but you can also delete items. Using:

remove value at will delete 1 item from your array. You just tell it the key or index where you want to delete it.

Can you print your blocks and then after each printing, delete the last item in the array? You should report your action to the user so they understand what happened. The beginning of this is posted in the picture. Try to fill in the rest.



Expereiments

EXPERIMENT 1 - PRINT BLOCKS AND NAMES… TWO ARRAYS IN PARALLEL

This prints the blocks in an array but there is a second array which holds the names of the blocks. By saying in parallel, it means the arrays have connected keys or indexes.

For example:

```
Blocks[1] = [159] BlockNames[1] = ["Terracotta"]
```

These two pieces of information are talking about the same thing. One array stores the ID and the other stores the name but they both have the same key. Having the same index makes it a lot easier to get to all the information at one time if we want to.

In theory you could make several arrays in parallel to store multiple pieces of information.

For example, maybe you wanted to create profiles for your monsters. Consider four arrays in parallel.

```
Name[1] = ["Bob"] FavoriteFood[1] = ["pizza"] FavoriteColor[1] = ["red"] Age[1] = [25]
```

Here by using the key of 1 you could get multiple pieces of information stored about just one monster.

EXPERIMENT 2 - PRINT WITH DATA VALUES

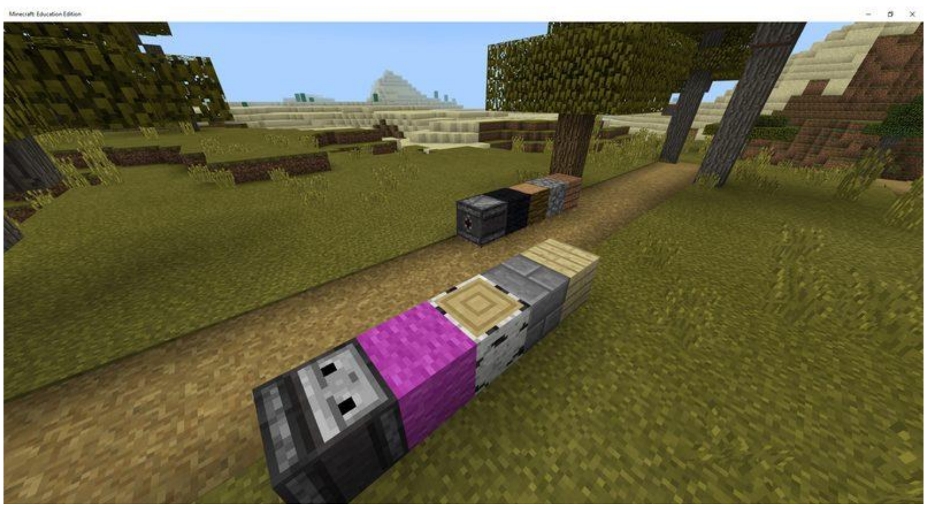
Some blocks or items from the game have a data value. The data value is used to print the same object but the style or color may vary depending on the data value. Think of it like having the same kind of car but with different options.

For instance, there are 16 types of concrete listed. All have the ID of 251 but each has a different data value (0-15, 16 total) to indicate a different color.

This code prints blocks from arrays but using the `DataValue` array to determine the style.

Because we use a for loop here we are able to access three different types of information about the same block, all with the same key!

This is very dangerous and the listings for data values did not seem accurate at the time of this publication. Again, experiment if you want to but be wary. It might be best to first find blocks by ID and then experiment by playing with data values. This would be the safest way to avoid a game ending error.



Activity: Quiz

Write your ideas here:

LESSON E: GET CREATIVE WITH ARRAYS

Today's activities

- Play a coding activity to experiment with arrays: Independent Coding Project

Coding activity: Independent project

Arrays for art

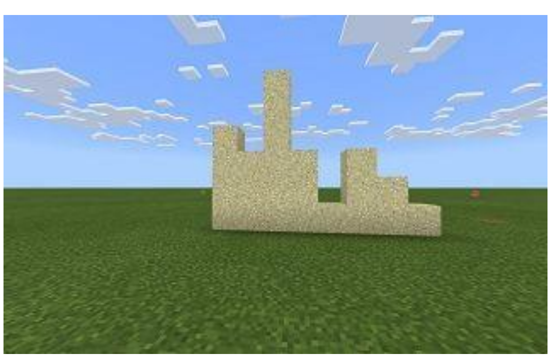
For this chapter's independent project, use an array to create a piece of artwork. For inspiration, take a look at the project called Rainbow Beacon in the Super Powers category in Code Builder. This project loads an array with colored wool, and then shoots it straight up toward the sky, as far as the eye can see.

How might you modify this project? Could you make a rainbow road stretching out to the horizon? Or create a flat square with a rainbow design? What about a rainbow house or tower? With arrays and loops, you have the power to create enormous public artwork that stretches out, and up, in different dimensions. See what kind of colorful, multi-dimensional artwork you can create!

Ideas

SOME IDEAS FOR ARRAY ART:

- Create designs and patterns by placing various types of decorated sandstone from an array.
- Create a maze of random blocks.
- Design intricate patterns in the ground so flying above them becomes fun.
- Create a pattern by using an array of only black and white wool blocks.
- Create a bar graph by dropping various amounts of sand from the sky based on the numbers in an array. The dropping would be interesting to watch!



Water creates some cool effects when placed outside a container.

- Lava and fire are also interesting to work with.
- Don't forget you can erase blocks by filling them with air.

Project criteria

Your project should do the following:

- Use an array of items.
- Access items in the array either in sequence or at random.
- Only access indexes that exist in the array (prevent "out-of-bounds" access).

"out-of-bounds" access

This means that it is possible to give an index that would cause an error.

For example, maybe my array has two items in it.

```
item1 = myitems[0] item2 = myitems[1]
```

If a user tried to access

```
myitems[4]
```

it would cause an "out-of-bounds" access error.

"out-of-bounds" prevention

if then else statements are a great way to check and prevent this kind of error.

Minecraft Diary

Expectations

Compose a diary entry addressing the following:

- What kind of art did you decide to make? What does your program do?
- Describe how your program creates its artwork.
- How did you ensure that only valid indexes are accessed?

NOTE: If you decided to improve one of this lesson's activities, please talk about the new code you wrote in addition to what was already provided in the lesson.

Grading criteria

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.
Array	Array is not created properly or used at all, no means of stopping out-of-bounds access was implemented.	Array is properly created, some elements not reachable, and out-of-bounds access not prevented.	Array is properly created, (possible to access all elements or no out-of bounds access)	Array is properly created, possible to access all elements, no out-of-bounds access.
Project ▪ Use an array of items. ▪ Access items in the array either in sequence or at random. ▪ Only access indexes that exist in the array (prevent "out-of-bounds" access).	Project lacks all three of the required elements.	Project lacks two out of the three required elements.	Project lacks one out of the three required elements.	Project creates a piece of artwork, efficiently and effectively.

GLOSSARY OF KEY TERMS

Arrays: An array is a series of places to store things. You can think of it like many variables in one place. You can store several items under the same name and you only need that one name to find the information. This information could be a list of items, a row of mailboxes, or a train of container boxes. for example. The information in the array is all similar.

Array Index: A unique address or location in the collection (for example, page number in an album, shelf on a bookcase, and so on)

Array Length: The total number of items in the collection

Array Sort: How you could order items in the collection (for example, date, price, name, color, and so on)

Array Type: The type of item being stored in the collection (for example, DC Comics, $1 coins, Pokémon cards, and so on)

Builder: The Builder is like an invisible cursor in the game that can place blocks along a path very quickly.