

Educator Guide

Unit 5 – 4 Lessons
60 minutes per lesson
Single Student

Coding with Minecraft: 5- Conditionals

EDUCATION.MINECRAFT.NET

THEME OVERVIEW

In this unit, students will explore the coding concept of conditionals. An important part of programming is telling the computer when to perform a certain task. Conditionals accomplish this task by requiring a certain condition or rule to be met before an action is performed. Students will have the opportunity to participate in unplugged activities, code programs using conditionals, and evaluate and improve existing code. Students will the participate in the final activity to use collaborative design and pair programming techniques to solve a problem players encounter in Minecraft.

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 Conditionals
- Lesson B: Coding with Conditionals (this lesson is broken into 2 sessions)
- Lesson C: Get Creative with Conditionals

LESSON OBJECTIVES

- Learn and apply the coding concept of conditionals
- Create IF THEN and IF ELSE THEN conditional statements
- Explain the importance of conditionals in coding and while playing Minecraft
- Evaluate code to identify problems and debug the problem code
- Work collaboratively to design an original creative project to apply their coding skills in new ways

THINGS TO KEEP IN MIND

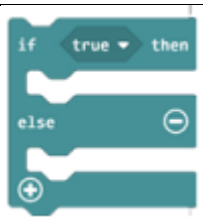
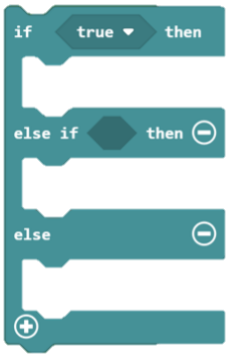
- Remind students that there may be more than one solution for each of the activities.
- Students should keep a coding mindset. As with any other sport or activity, you must practice in order to get better. Coding is no different.

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

	**On chat command** Runs the code when the student types the chosen text in the chat window
	**If** Run code depending on whether a condition is true or false
	**Else** Runs a specific code depending on whether a condition is met or not met
	**Else if** Another conditional action to add an if to an else for an else if
	**While** Repeats the code is a defined condition is met

KEY VOCABULARY

Conditional – if a certain condition or rule must be met before an action is performed

LESSON ACTIVITIES

Lesson A: Introduction to Conditionals (Slides 1-28)

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 conditionals. (**slide 1**)

Today, we are going to learn about what are conditionals and how we use them not only in real-life, but how they are used in Minecraft. (**slide 2**)

Review the lesson objectives. (**slide 3**)

So exactly what is a conditional? **(slide 4)**
Computer programs are made up of instructions that tell the computer how to process input and deliver output. In Minecraft, the instructions you write are triggered by events or by commands, and they control what actions take place in the Minecraft world. An important part of programming is telling the computer WHEN to perform a certain task. For this, we use something called 'conditionals', because a certain Condition or Rule has to be met before an action is performed.

Students are already familiar with the concept of conditionals in their daily lives. Have they ever had a parent say any of the following?

- "If you clean your room, you can go out with your friends." (**slide 5**)
- "If your homework is done, you can play video games." (**slide 6**)

These are all conditionals! Conditionals follow the format of **IF** this, **THEN** that.

IF (condition is met), **THEN** (action performed)

There are also IF THEN ELSE statements: **IF** (condition is met), **THEN** (action performed), or **ELSE** (action performed).

Example: (**slide 8**)
IF it is snowing, **THEN** wear boots, **ELSE** wear shoes.

The **ELSE** portion makes sure that some action is performed regardless. Regardless of whether it is snowing or not, you should probably wear something on your feet.

For the “IF it is snowing, THEN wear boots, ELSE wear shoes” conditional, notice that without the **ELSE** action, your students might end up not wearing any footwear!

Have the students share a few conditionals from their own lives with the class or within small groups.

Guided Instruction (Teacher Modeling; “We Do”)

Unplugged Activity: Simon Says IF ELSE THEN (slide 9)

Purpose: This is a quick variation on the Simon Says game to practice conditionals (IF, THEN, ELSE statements)

How to play

- Everyone stands up
- The teacher will say a series of conditional statements – some may just be If… Then…, but some may be IF… Then… Else.
- Students will follow the instructions

Here’s a simple example:
The teacher will say “**IF** you have brown hair, **THEN** raise your right hand, **ELSE** sit down.” So, everyone who has brown hair should have their right hands raised, and everyone else should be sitting down.

Teacher tips

- Lead the activity to practice conditionals with simple instructions, yet visually different physical poses.
- Verify that students have correctly executed the instructions based on the criteria.
- Read the conditional statements slowly and clearly.
- Make sure students understand and are following along with each set of statements.
- Reset the students to normal standing positions between each conditional statement.
- Ask the class for a few volunteers to create their own conditional statements for the rest of the class to follow along.

Examples of conditional statements

- **IF** your name starts with a ‘J’, **THEN** give me a thumbs up

- **IF** the month of your birthday ends with a 'Y' or an 'R', **THEN** raise both your arms
- **IF** you play soccer, **THEN** kick your feet (be careful not to kick anyone)
- **IF** your favorite ice cream flavor is chocolate, **THEN** stick out your tongue, **ELSE** make rabbit ears behind your head
- **IF** you play a musical instrument, **THEN** snap your fingers, **ELSE** whistle
- **IF** your favorite monster in Minecraft is a zombie, **THEN** make zombie noises and movements, **ELSE** jump up and down

Other example conditions

- Your hair is brown / blond / red
- Your eyes are brown / blue / green
- You are wearing a sweater / shorts / T-shirt / pants / skirt / sneakers / sandals
- It is raining / sunny / cloudy today
- You've played soccer / volleyball / football / frisbee in the past month
- You've eaten spaghetti / cereal / ham / toast in the past week
- You slept well last night / had a hard time waking up this morning
- Your favorite ice cream flavor is chocolate / vanilla / strawberry / mint / cookie dough
- You have a pet dog / cat / bird
- You have a sister / brother / no siblings

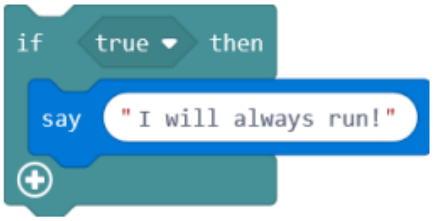
Types of conditionals in MakeCode (slide 10)

MakeCode for Minecraft features a conditional block called an **IF… THEN…** block.
There is also another related block called the **IF… THEN… ELSE…** block.
These blocks are in the LOGIC Toolbox drawer. These blocks will check to see if a certain condition is true, and if so, then they will perform their operations.
If you click the little plus sign at the bottom of these blocks, you can add more conditions.



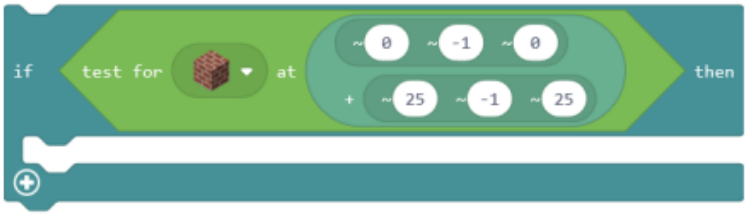
Examples (slide 11)

By default, IF statements have a hexagon that says “true” in them when you first place them. If you don’t change this, then anything that the “If… Then…” block encloses will run every time. “True” is always true.

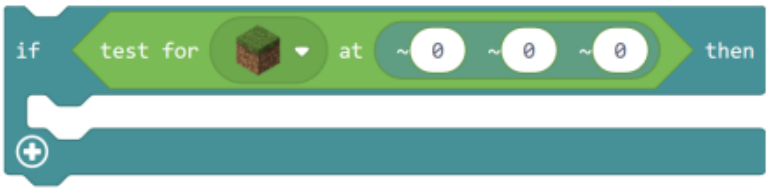


However, you can substitute another block, or a combination of blocks, next to the If. The blocks you use as conditions do not have to be hexagons but ultimately should resolve to true or false.

This example from the BLOCKS Drawer tests for a certain type of block at a specific set of coordinates.



In this example, they represent the player's exact location. This is always going to be false because the player is standing there, so there cannot be a block where its head or body is. However, you can specify **~0 -1 ~0** to check the block the player is currently walking on, or even use a Position range to detect the presence of a certain block within a given area.



In this chapter, you'll explore some of the possibilities that the different conditional blocks provide, by automating some common Minecraft tasks like chopping down trees or mining. You'll also be building things like pyramids and making your code work a bit more intelligently so it responds to what the user inputs. Then you'll challenge yourself to create your own independent project automating some other Minecraft task of your choice. Let's go!

Coding Activity: How Old Are You? (slide 13)
Import the Program:
https://minecraft.makecode.com/?ipc=1#tutorial:github:mojang/educationcontent/CodingWithMinecraft/unit_5/lesson_A

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 14**)

Once in the Coding with Minecraft subject kit, select Unit 5: Conditionals to begin the lesson. (**slide 15**)

In this programming activity, students will practice using the Conditional if then else block in MakeCode. They will code a program that uses a condition to compare their friend's age with their own, and print different messages if they are younger, older, or the same age. **(slide 16)**

Steps for the Coding Activity: How Old Are You? **(slides 17-24)**

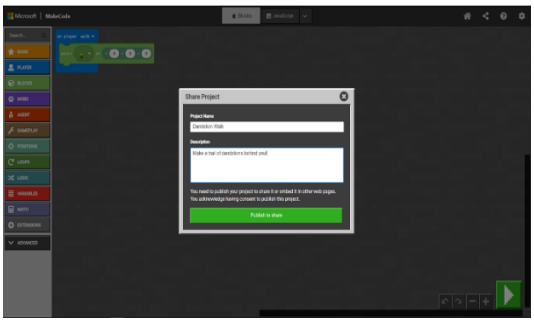
- **Step 1**: Rename the existing on chat command "run" to "age".
  Click the Plus **(+)** sign on the on chat command "age" to create a num1 variable parameter.
  Rename **num1** to **FriendsAge**. This makes your code more readable. Always use meaningful names when making variables in coding. This makes finding errors more intuitive. You can rename variables from the drop-down menu.
- **Step 2**: From LOGIC, drag a if then else inside on chat command "age". Click the Plus **(+)** sign in the if then else to create another else if branch.
- **Step 3**: From LOGIC, drag a Less than, 0 < 0, comparison block into the if slot, replacing true.
- **Step 4:** Again from LOGIC, drag an Equals, 0 = 0, comparison block into the else if slot.
- **Step 5**: From the VARIABLES Toolbox drawer, drag two of the FriendsAge blocks into the first slot of each of the Comparison blocks. In the second slot of the Comparison blocks, enter your age (for example, 12).
- Step 6: From BLOCKS, drag a print "Hello" onto your workspace. Then you can right-click print "Hello" and select **Duplicate** to make a copy. Place one in each of your if, else if, and else clauses.
- **Step 7**: In each print "Hello", type a different message for people who are younger than you, your same age, or older than you. Use the drop-down menu in the print blocks to select a different block to use for each message.
- **Step 8**: Change the grass blocks to whatever block you like. Then, in all the print blocks, set the **Y** coordinate to 10 (so these messages print in the sky).

Run and test the program for How Old You Are? after you 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.

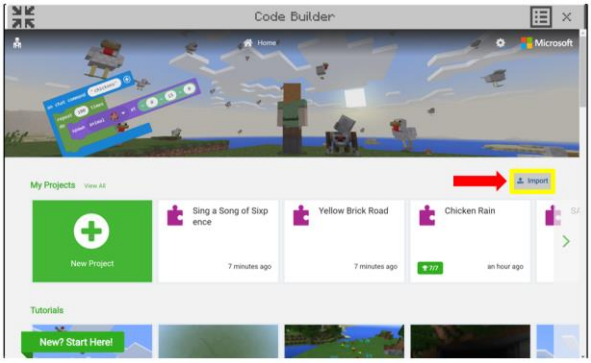


Step 3: Select the green "Copy link" button.

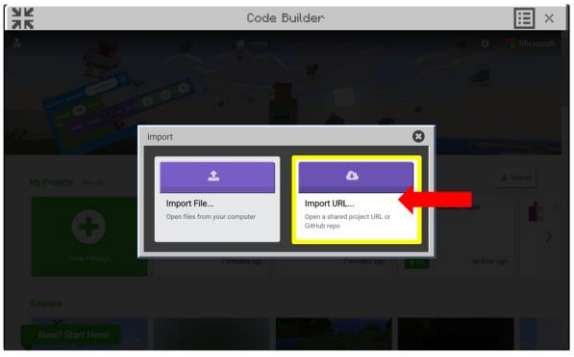


NOTE: If you are having difficulty with the copy link, try using the short cut: press Ctrl + C (or Command + C) on your device.

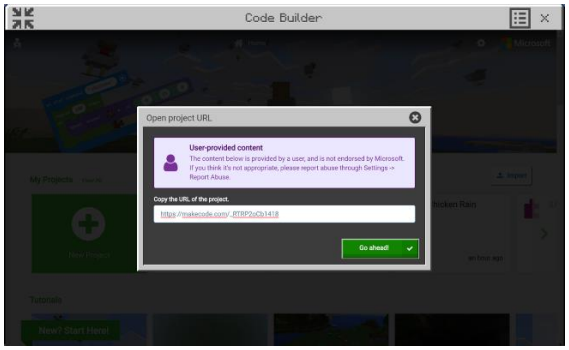
Step 4: Inside Code Builder, select the "Import" button.



Step 5: Select the "Import URL" button.



Step 6: Paste the Project URL. Then, select the "Go ahead!" button.

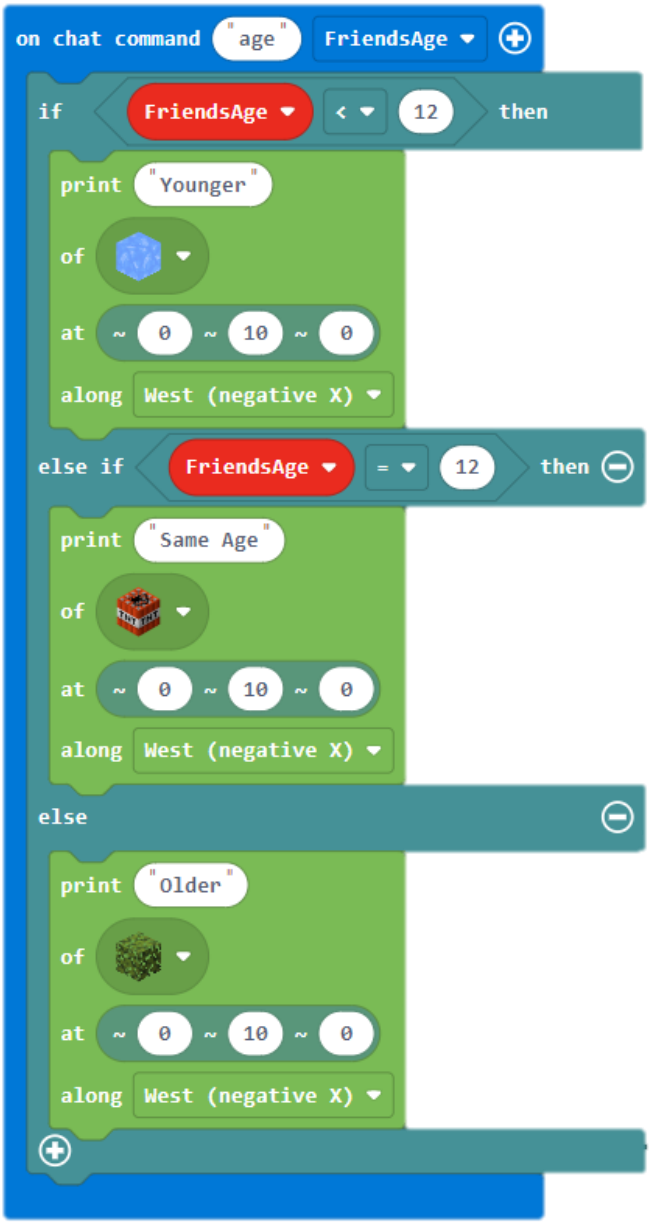


NOTE: If you used the Ctrl + C (or Command + C for a Mac) to copy the link, you would then paste by pressing Ctrl + V (or Command + V) to paste.

Press "t" to open the chat window. Enter "Age" and a number into the chat window. For example, if we type "Age 2"—the word younger should be spelled in

the sky. Why? Our friend is 2 years old, therefore they are **younger** than us (we are age 12). **(slide 21)**

Coding Solution:



Independent Work (Teacher Support; "You Do")

Students should take the time to make their own unique and different coding modifications through one of the challenge activities:

Challenges

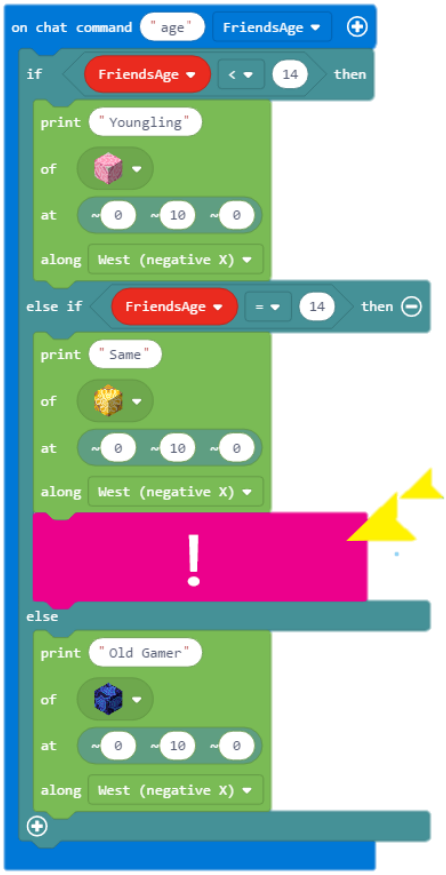
Challenge 1 - Compare Ages and Modify Messages

Rather than compare students at 12 years old, compare students to another age. If you're brave, ask your teacher how old they are and put their age in for comparison. Modify the messages for people who are younger than your new age, the same age, or older than your new age. Finally, modify the blocks that print.

Challenge 2 - Compare Birthdays

Place in another if then else to check whether your friend has the same birthday as you.

To do this you will need to take in another variable from on chat command "age". Using a number variable might be easiest, but you could do this in other ways too. In the following picture, you can see where you would need to test for the proper month, so put your new if then else there.



Extensions

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

- **Extension 1 - Try Other Types of Conversions**
  Use this extension to change your weight from pounds (lbs) to kilograms (kg) and vice versa.
  - Use this code to start with.

    - If you enter weight in the chat window, the code shows you a menu that explains the program.
    - If Statements allow you to detect what the user wants to convert to. Also, an If Statement checks whether it is ok to display an answer message.
    - How could you change this code so that the output is not in the chat window? Maybe you could print blocks instead as an answer… What other things could you ask people?
- **Extension 2: Create Blocks and Items**
    - This code has a menu. Enter create in a chat window.
    - The code accepts the ID for blocks and items.
    - If the ID is for a block (1 - 255), the block is placed.
    - If the ID is for an item (256 - 452), the item is equipped to your character.

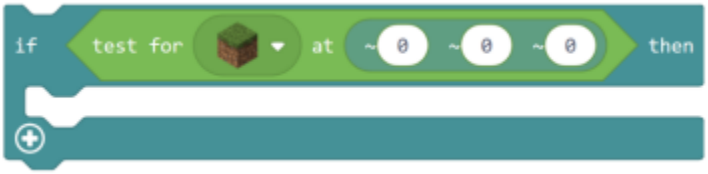
LESSON CONCLUSION

Upon completion of this lesson, students should be able to answer the following questions: (**slide 26**)

1. In MakeCode, the conditionals blocks are found in which toolbox drawer?
   Answer: Logic
2. What are some of the types of conditional blocks?
   Answer: If then, if then else, comparison, Boolean

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

1. Explain what this code means:



   Answer: A conditional that tests for a certain types of block (grass) at a specific set of coordinates (i.e., the player's exact location)
2. What is an example of an IF THEN or IF THEN ELSE statement?
   Answer: Responses will vary and need to follow one of the following formats:
   IF _____, THEN _____
   IF _____, THEN _____, ELSE _____

Conclude the lesson by reviewing the concepts covered in Lesson A. (**slide 27**)

Provide a preview of the upcoming lesson. (**slide 28**)

Lesson B: Coding with Conditionals (Slides 29-47)
*This lesson will take 2 sessions (i.e., 45-60 minutes each)

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

Welcome back to Coding with Minecraft. We are going to continue to learn about and practice using conditionals in our coding. (**slide 29**)

Review the lesson objectives. (**slide 30**)

Chopping trees for wood is hard work but necessary in Minecraft if you want to craft objects and tools in Survival mode. But you can automate this chore through code with the help of our Agent! Let's teach the Agent how to chop down a tree of any height and return back down to the ground. (**slide 31**)

Guided Instruction (Teacher Modeling; "We Do")

Coding Activity: Chopping Trees (slide 32)
Conditionals Tutorial World
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.

The last activity used if then else statements to check conditions and perform actions the way you wanted them done. In this activity, you will use conditionals in a while loop. The agent will continue moving up as long as there is a block of tree to climb. (**slide 33**)

Have all students log into Minecraft: Education Edition. Students will need to navigate to the coding tutorials from the in-game library. Directions for navigating to the tutorials can be found on **slide 34**.

Once students log into the world, they will begin game play at their spawn point. (**slide 35**)

Students will need to talk to the Intermediate NPC. **(slide 36)**

Students should select “Blocks” as their programming language. **(slide 37)**

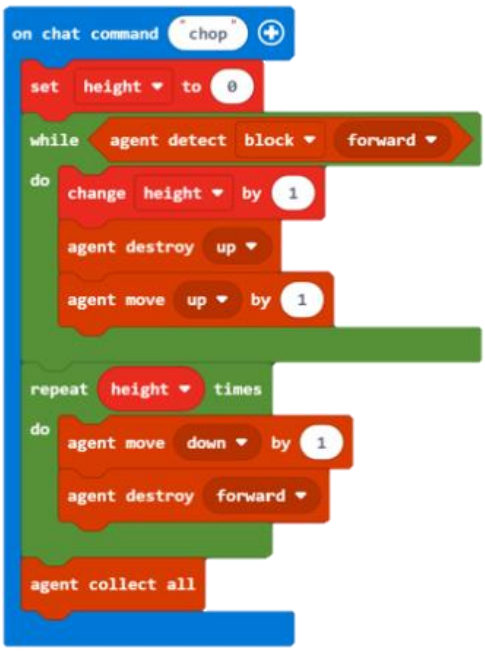
They will respawn into a new area. They will see a group of trees in front of them. Select “C” to begin coding. **(slide 38)**

You should demonstrate each step of the coding activity alongside students. Model each individual step. **(slide 39)**

After the code is complete, your Agent will spawn onto the gold block. However, in order for the Agent to successfully chop down the tree, they must be standing directly in front of the tree. You will need to position the Agent directly in front of the tree before running the code. You (the player) need to stand in the exact location (and direction) of where you want the Agent to teleport to. Once you are standing in the spot, open the chat window and type “tp”. Your Agent will then teleport to this exact spot. **(slide 40)**

Then, you should run and test your code to see if the Agent successfully chops down a tree! (**slide 41**)
Students can continue to practice by chopping down the rest of trees in the row.

Coding Solution:



Extensions

Extension 1 – Increase the Size of a Hole

The Agent works hard for you! You can get it to dig for you too. Enter the on chat command dig 4 to try out this code.

Can you get the Agent to dig and place something at the same time? Can you increase the size of the hole? Right now, it is 2 x 2. You might get the Agent to dig a 4 x 4 hole and line it with gold as it goes down.



Experiment 2 - Create a Maze

This experiment code is aimed at getting you to think about how you might build some code to create a tunneling robot. The Agent will accept two numbers this time. First, it digs down and then will dig horizontally.

Enter the on chat command dig 5 4 to try out this code.

How could you get your Agent to build an underground maze? What else might you do with this underground hole it is digging? You could fill it with lava or put some mobs down there or even build an underground home. What other pieces of code might help you do this?



Independent Work (Teacher Support; "You Do")

Coding Activity: All Mine (slide 40)

Import the Program:

https://minecraft.makecode.com/?ipc=1#tutorial:github:mojang/educationcontent/CodingWithMinecraft/unit_5/lesson_B/all_mine

In addition to chopping trees for wood, you will probably want to mine the earth for precious minerals. One good strategy for finding the rarest of minerals, diamond, is to tunnel deep beneath the earth to levels Y12 or Y13, and then create a series of branching tunnels off a central passageway.

Getting down there!

Find your world position in the game (press the F1 key to see your world position on the screen), and then enter the command: /tp X 12 Z, where X and Z are your current world coordinates. This will teleport you directly down to level 12 altitude in your Minecraft world. Make sure you equip yourself with a pickaxe and torches to make some room and to see down there!

In this activity, you'll create a basic mining agent that automatically destroys blocks in front of it and collects everything. You'll use conditional statements to test for precious minerals. The agent is one block tall, so you'll send it forward 64 blocks, and then have it turn around, move up one block, and come back. If the agent finds anything of value, it will let you know and collect the ore.

Do the activity

Create controls for the Agent

1. Create a new MakeCode project and name it **All Mine**.
2. Rename the existing on chat command to "do".

Because you are always doing things with your agent, you should make it easy to command it. You can do this with a if then else statement. You can make a simple menu to handle all the agent commands.

3. Drag a if then else into on chat command "do". Click the **(+)** sign on the if then else to add a third branch for this because you will test for three conditions.
4. Click the **(+)** sign on on chat command "do" and rename num1 to **AgentOrder**.

Now just test to see what the user entered. You want the user to be able to teleport the agent and turn it. The third branch will give the user some information if nothing is entered.

5. From LOGIC, grab the ‘0 = 0’ comparison block and duplicate this because you need two.
6. Place these into the if then else.
7. Adjust these so you test for what the user entered. We need to test the value of AgentOrder. Let’s say a value of 1 means the agent teleports and a value of 2 means the agent should turn.

Last, you just need to add blocks to the branches to make your agent do the actions.

8. From AGENT, add agent teleport to player into the first branch of your if then else.
9. From AGENT, add agent turn into the second branch of your if then else.
10. In the last branch, add a message to give the user directions. From PLAYER, add a say to the third branch of your if then else.
11. Adjust the text in the say to read **Enter 1 to teleport or 2 to turn**.

There, now controlling the agent is a lot easier.

Make the *“dig”* command

12. From PLAYER, place a on chat command into the coding Workspace, and rename it to "dig".

You can only mine diamonds in **Survival** mode, so the first thing you should do is make sure you change the game mode to Survival.

13. From GAMEPLAY, place change game mode into on chat command "dig".
14. Adjust change game mode by clicking the player selector drop-down menu to select yourself @s.

From LOGIC, drag a if then else out and drop it after the change game mode block.

Look for diamonds

You want to give an alert if the agent finds valuable blocks like diamond ore.

15. From LOGIC, grab an Equals (=) comparison block to replace the true in your if then else.

16. From AGENT, drag inspect into the first slot of the Equals, 0 = 0, replacing the number 0. The inspect block will check the block directly in front of the agent.
17. From BLOCKS, drag a block out and drop it into the second slot of the Equals, 0 = 0.
18. In block, use the drop-down menu to select Diamond Ore as the block you are checking. (You can also use the Search bar to find it.)

Get rich!

If you do find Diamond Ore, then you'll want to print a message to the screen and mine the ore.

19. From PLAYER, drag say and drop it into the first branch of the if then else.
20. In the say block, enter a message like "We're rich!".
21. From AGENT, drag the following three blocks and drop them in order under the say block: agent destroy, agent move, and agent collect all.

Destroy the other blocks and keep mining

If the agent doesn't find Diamond Ore, he should simply destroy the non-valuable block in front of it, and move forward without collecting anything.

18. From AGENT, drag agent destroy and agent move into the else clause.
19. You need to repeat this process 64 times, so you'll use a loop. From LOOPS, drag a repeat loop out and surround your if then else block.
20. In the repeat loop, enter the number **64**.

Make a return trip

This will send your agent forward to mine 64 blocks, but you need to have it turn around and come back. So, you'll use another loop outside your repeat loop. You are creating a "Nested Loop" because there will be a loop within a loop.

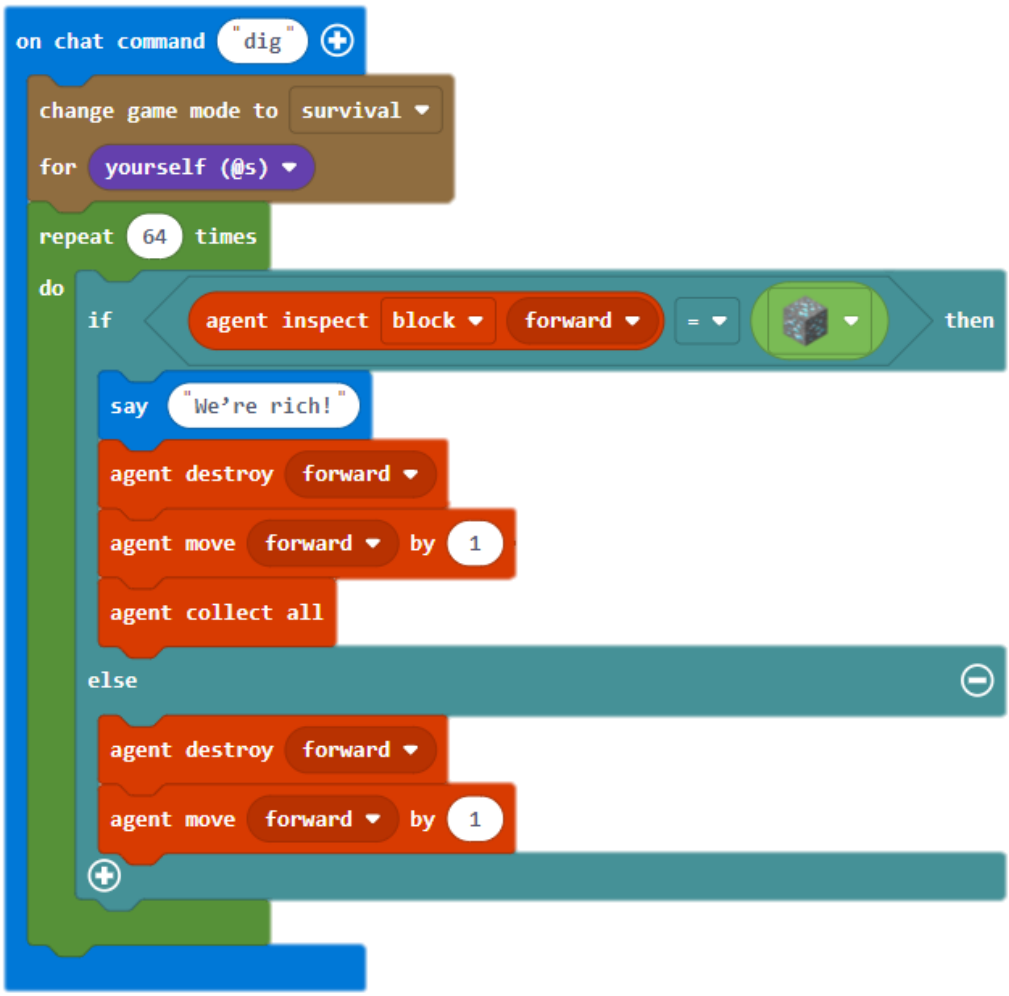
21. From LOOPS, drag another repeat loop out and surround your existing repeat 64 loop.
22. In the repeat loop, enter the number **2**, because you'll want your agent to mine just two rows of blocks. At the end of the first row, you want your agent to move up and turn around.
23. From AGENT, drag the following four blocks out and drop them in order under the inner repeat loop: agent destroy, agent move, and two agent turn blocks. You place them here to position your agent to come back. You will need to adjust the direction for a few of these blocks by using the the drop-down menu to select up.

Getting down there!

Remember that you will first need to teleport your character down to where you can mine.

Find your world position in the game (press the F1 key to see your world position on the screen), and then enter the command: /tp X 12 Z where **X** and **Z** are your current world coordinates. This will teleport you directly down to level 12 altitude in your Minecraft world. Make sure you equip yourself with a pickaxe and torches to make some room and to see down there!

Coding Solution



Lots that could go wrong!

Several factors are at play when mining deep under the earth! You might teleport down and find something unexpected.



The agent might not respond the way you initially think it should. The agent will not dig unless it is near your character when the command is issued, so you might get a message like this:



Remember that sometimes restarting your code can be helpful if you find things are not working as they should.

Challenges

Let’s set up a on chat command that will check for two conditions. If the user enters **1**, the character will be teleported down to level 12 automatically. If the user enters **2**, the user will be teleported out of the ground.

Challenge 1 - Set Up Conditions

Set up an on chat command "go". Make this so on chat command "go" accepts a variable called **WhereTo**. Put an if then else inside of go.
Test for these conditions:

1. User enters “1” (down)
2. User enters “2” (up)
3. User enters nothing

You will need to compare what the user enters and then add the following to the if then else.
Put the following inside the up branch or 1:
Put the following inside the down branch or 2:
Give the user a message if they enter nothing.

Challenge 2 - Add More Conditions

What else could you do to make on chat command "go" from Challenge 1 better?
Options:

- What about equipping your character with torches automatically before going down?
- You might clear out a small room so that after the character teleports they have some space.
- You could automatically teleport the agent to your character after you go down to level 12.

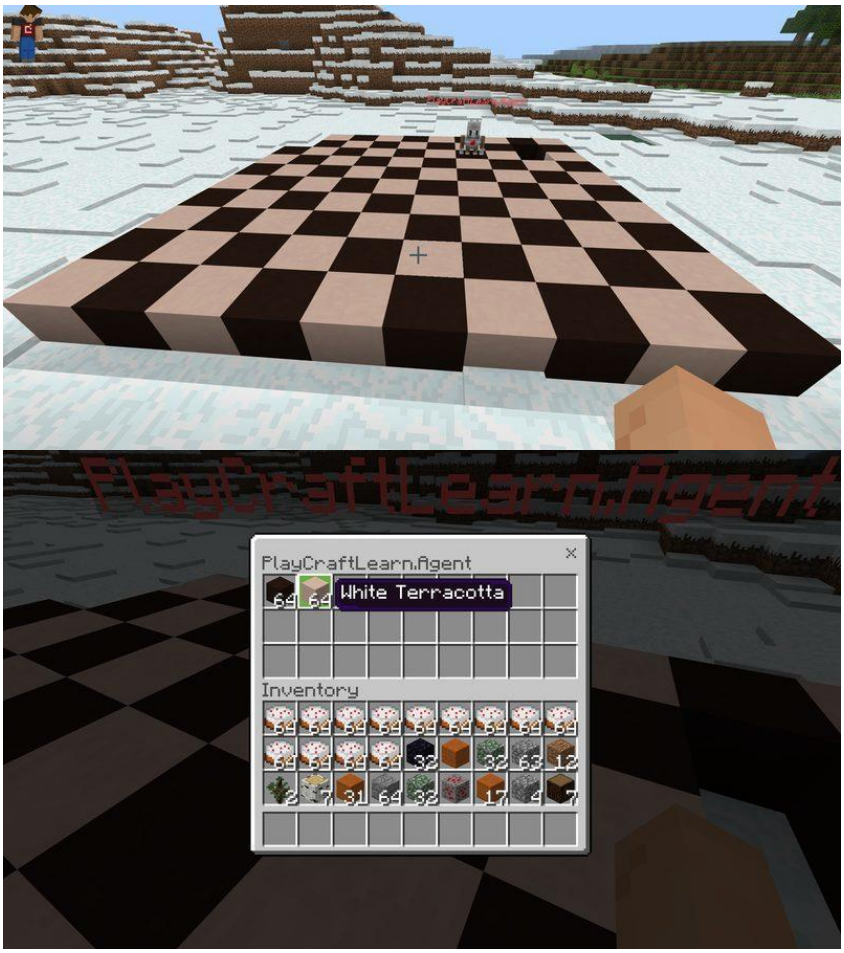
Choose at least one of the options above or your own modification and implement your option(s) for this challenge.
Note: The solution has all three preceding options included in it for demonstration.

Experiments

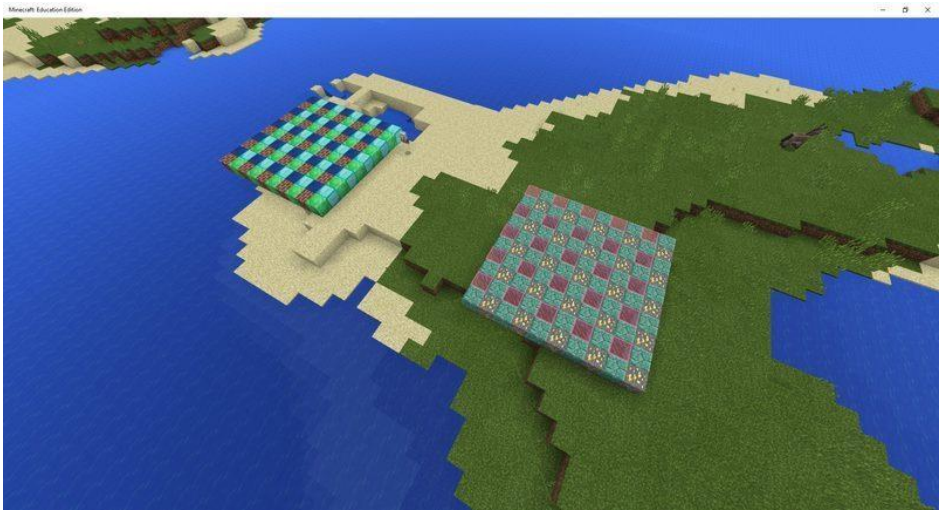
Here, there are no rules… Copy the code for each experiment and change things around to see what kind of results you can create. Suggestions are given, but do as you like!

Experiment 1 - Standard Checkerboard

Have your agent build a checkerboard. Make sure slot 1 and 2 have blocks in the agent’s inventory!



This uses a bit of math and a if then else to alternate blocks and give that checkerboard look.
Use this [code] to start with.
How could you change this? Could you make other patterns by playing with the if then else?



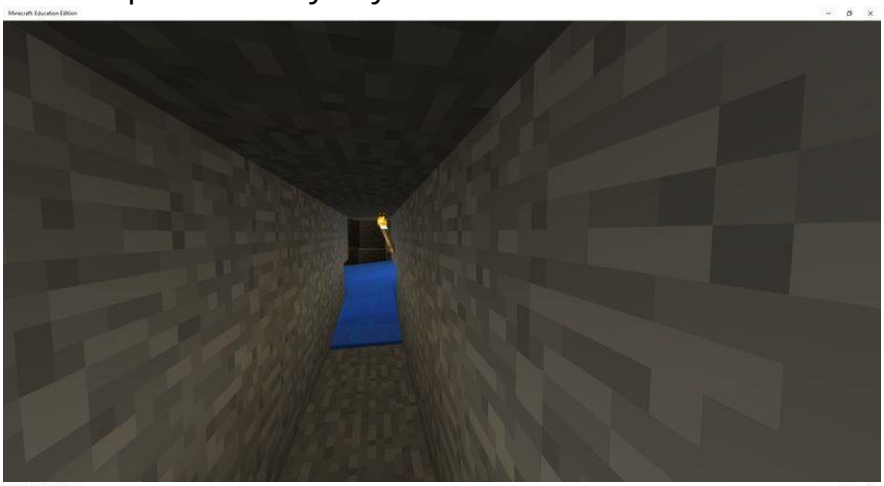
Could you add ideas from other code you have seen to make this different and more interesting? What about one block up and the next down?

Experiment 2 - Maze Digger

This is similar to an earlier activity. Here the agent will dig for you BUT it will generate a random maze. There are several concepts we have discussed as well as a new one, functions. Functions will be covered in the next lesson, so you should take a look at them.

The agent turns randomly with the help of a if then else that checks what random number is generated.

Use this code to start with.
How could you make this even more random? A cool maze definitely needs to be random! The natural features of the world will help, but what else could make this maze super fun? Maybe you could add animals or monsters as well!



LESSON CONCLUSION

Upon completion of this lesson, students should be able to answer the following questions: (slide 44)

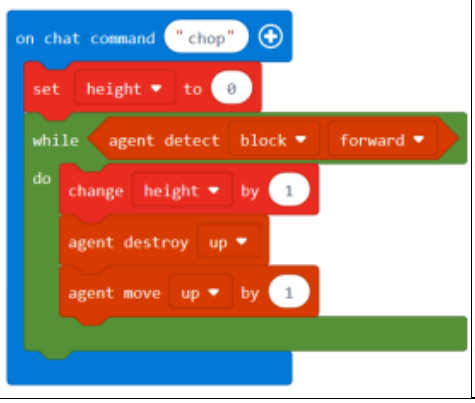
1. What is the other block that can be used for conditionals but isn't located in the Logic toolbox drawer?
   Answer: While loop block
2. What are two ways to restart your code?
   Answer: Use the stop button or resave your project

Conclude the lesson by reviewing the concepts covered in Lesson B. (slide 45)

Provide a preview of the upcoming lesson. (slide 46)

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.

Questions	Answers
1. What is an example of an IF THEN statement?	Responses will vary, but they need to follow the format: IF _____, THEN _____
2. What's an example of an IF THEN ELSE statement?	Responses will vary, but they need to follow the format: IF _____, THEN _____, ELSE _____
3. In coding, conditionals tell a computer to do which of the following? a. How to do it b. To do it faster c. When to do it d. Where to do it	b. When to do it
4. What type of blocks acts like a like a conditional but isn't located in the LOGIC toolbox?	A while loop
5. What's a reason you might need to restart your code? a. Things aren't working as they should	d. all of the above

b. An infinite loop c. You get the message "Cannot issue command, Agent is out of range" d. All of the above e. Only a and c	
6. Using words, describe what the following coding blocks do: 	Something like: • When you type chop in the chat window • And your Agent detects a block in front of it • The height variable increases by one block, • The Agent destroys the block above it • And moves up by one block

Then provide students with a preview of the next lesson. **(slide 60)**

Lesson C: Get Creative with Conditionals (Slides 47-67)

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 conditionals. (**slide 47**)

Review the lesson objectives. (**slide 48**)

Guided Instruction (Teacher Modeling; "We Do")

Paired Independent Project

In this chapter, you've looked at how some of the different conditional blocks work. You've seen that a while loop is like a repeat block combined with a conditional statement (repeat while this condition is true), and you've also seen how you can use a series of If statements separated by Else statements to create blocks of code that run in all sorts of different cases. Finally, you've looked at Boolean operators, which define a relationship between conditions such that both

conditions must be true, or only one of the conditions needs to be true, in order for the whole expression to be considered true.

For the independent project, work with another student to create a project together that uses one or more blocks from the Logic Toolbox drawer. Think about problems you encounter in a Minecraft world and try to come up with solutions for them. (slide 49)

Paired Project Work Techniques (slide 50)

Collaborative Design

In any design project, it's important to start by understanding the problem. You can begin this activity by interviewing people around you who might have encountered the problem you are trying to solve. For example, if you are designing a survival tool, what do most Minecraft players do to survive the first night? Ask your friends how they find food, build a shelter, and so on. What resources do they tend to look for first? Do they have a system? What would be more effective? What do they wish they had the ability to do?

If you are designing something else, think about how you might find out more information about your problem through interviewing or observing people playing Minecraft, with your focus being strictly on studying their behavior in-game. A good way to focus your observations is to make a list of key questions ahead of time:

- What do new players do first? How quickly do they decide to go underground?
- How long do they stay there?
- Do they tend to stay in one place or migrate?
- Which characteristics make a good place to build a base?

Then start brainstorming with your partner. Talk about a variety of different ideas. Remember that it's okay if the ideas seem far-out or impractical. Some of the best programs come out of seemingly outlandish ideas that can ultimately be worked into a useful program.

Pair Programming

The concept of "pair programming" is a valuable way to have programmers collaborate when creating programs together. Two programmers share one computer, with one person at the keyboard acting as the driver and the other person providing directions as the navigator. If you are collaborating with someone else on a project, try using one computer and take turns creating code

and then seeing how it behaves in the Minecraft world. The main rule is, if you are at the keyboard, you are doing what the other person is telling you to do. They are the navigator! Remember to practice good communication with each other throughout the entire programming process.

Debugging

Be sure to test out a number of different ideas in Minecraft. What is the easiest way to keep track of data? You can test to see if certain conditions are actually being detected by adding a Say block inside the If Then block. If you do something in the world that should trigger the condition, but you don't see the message printed to the screen, then you know that something is wrong with your condition. Similarly, if you are updating your variables based on certain conditions, you can use a Say block to print out the value of the variable so you can make sure it is changing properly. If you are using a number variable, remember to use a Join block to join it with a word (or empty quotes) so it will print.

Project examples (slide 51)

Here are some examples of challenges for your Independent Project:

Iron Finder

One of the most common tasks early in the game is finding enough iron to make iron tools and perhaps a full set of iron armor for protection. Iron is found in streaks of iron in the stone hillsides and caverns, but it is often hidden from view. Create a MakeCode Project that checks to see whether iron is near you, and then alerts you when you are very close to iron ore. After you find one block of iron ore, more is likely close by, so all you need is to hit on one block to strike an iron vein!

Improved Miner

Work with the agent program you defined and see if you can make it better. Can you make the agent check multiple levels at a time? Can you check for other rare minerals as well, like emerald?

Independent Work (Teacher Support; "You Do")

Minecraft Diary

Compose a diary entry addressing the following:

- What Minecraft problem did you decide to solve? What does your program do?
- How did you use conditional statements in your project?
- Discuss one (or more) ways that working with a partner was different from just doing the project by yourself.

- Describe one point where you got stuck. Then discuss how you figured it out.
- Include at least one screenshot of your agent in action.
- Share your project to the web and include the URL here.

Assessment
Competency scores: 4, 3, 2, 1

Diary
4 = Minecraft Diary addresses all prompts.
3 = Minecraft Diary entry is missing 1 of the required prompts.
2 = Minecraft Diary entry is missing 2 or 3 of the required prompts.
1 = Minecraft Diary entry is missing 4 or more of the required prompts.

Logic
4 = Uses conditional statements effectively in a way that is integral to the program.
3 = Uses conditional statements in a way that is integral to the program but some problems with conditional statement execution.
2 = Uses conditional statements effectively but in a superficial way.
1 = Doesn't use conditional statements at all or uses loops in a superficial way and has problems with conditional statement execution.

Project
4 = Project solves a specific problem, efficiently and effectively..
3 = Project lacks 1 of the required elements.
2 = Project lacks 2 of the required elements.
1 = Project lacks all of the required elements.

LESSON CONCLUSION

Upon completion of this lesson, review the concepts covered in this lesson. **(slide 53)**.

Preview the upcoming unit—Functions. **(slide 54)**

EDUCATIONAL STANDARDS

CSTA Standards

- 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