



MINECRAFT

EDUCATION EDITION

Educator Guide

2-3 Hours

Multiplayer

Seymour Island

[EDUCATION.MINECRAFT.NET](https://education.minecraft.net)

Causeway Digital

INTRODUCTION

Seymour Island is a multiplayer supported, “open world” style adventure, featuring a series of 11 coding tasks for students to complete. The world was designed to be completed with the full class of up to 29 students (plus 1 teacher), playing within the same shared world.

Throughout, students earn points for each task they complete, which allows for progress to be tracked. Although the world can be completed by a single student, it is highly recommended that the world is played with 2 or more students, as task 6-8 require a second player to help.

As this is an “open world” style activity, beyond the initial tutorial section (task 0), students are free to complete the available tasks in any order they wish.

The world makes use of the MakeCode platform for coding. Each task (except for task 0) is available both in MakeCode Blocks and Python. This gives students the option to try out and explore either language, depending on their preference or coding level. In addition to the coding tasks, there is also an addition cryptography/code cracking side quest involving a group of hidden robots.

Finally, it is advised to allocate students dedicated “partner” or “buddy”. This is because tasks 6-8 require an additional student to complete.

MATERIALS

Hardware	• The teacher will need a laptop with a projector for the plugged-in coding lessons. The students will connect to this laptop as it is a multiplayer world. • Internet access will be required for a portion of the lessons and activities. • Each student will need a device to complete the plugged-in coding activities.
Software	• Minecraft: Education Edition needs to be deployed on the devices utilized within these lessons. Use this link to find information about Deploying Minecraft: Education Edition. • The teacher will need the relevant Minecraft world files downloaded for the lesson. The students will not need the world downloaded as it is hosted by the teacher.
Other Materials	• Minecraft Education Edition Tutorials (if needed) • Minecraft Education Edition Supplemental Lessons (if needed) • Microsoft MakeCode References for Minecraft • Microsoft Python Notebooks Reference Guide



LESSON OBJECTIVES

- Develop programs with sequences and simple loops, to express ideas or address a problem.
- Break down the steps needed to solve a problem into a precise sequence of instructions.
- Develop plans that describe a program's sequence of events, goals, and expected outcomes.
- Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.

STUDENT PREREQUISITES

The following is assumed of students attempting the world

- They can move around the game, including using the needed controls for the device (i.e., keyboard & mouse or touch).
- It is recommended that students experience a [tutorial](#) prior to playing Seymour Islands.
- Although the world can be completed by a single student, it is highly recommended that the world is played with a minimum of 2 students, as task 6-8 require a second player to help.

THINGS TO KEEP IN MIND

- In each task, students are given a Blaze Rod, which they will use to execute the code they have written. This is instead of the usual Play button.
- Remind students there may be more than one solution for each of the activities.

DECIDING THE CODING ENVIRONMENT

Seymour Island includes support for student to complete each task in a block-based coding environment (MakeCode) or a Python based coding environment (Python Notebooks).

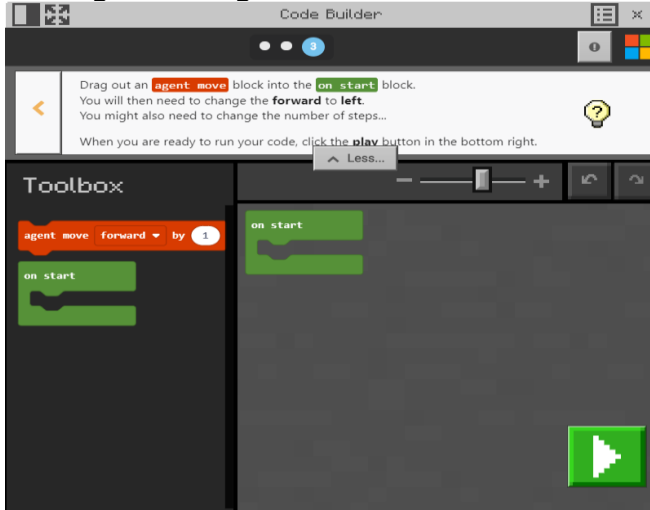
This choice is presented to students at the start of each task, allowing switching back and forth throughout their time in the world.



MAKECODE

MakeCode is a block-based coding environment suited to students new to coding or that don't yet have experience with text-based coding.

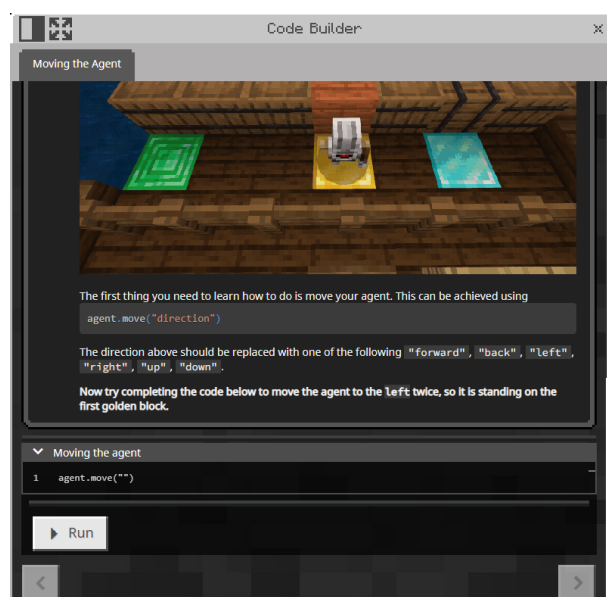
The MakeCode environment includes a step by step set up instructions at the top that can be toggled with the arrows, a toolbox on the left-hand side with only the blocks that are needed within the task and a coding box. Some tasks also include optional hints that can be displayed by clicking on the light bulb.



For an in-depth explanation of each of the blocks available within Makecode, a [reference guide](#) is available.

PYTHON NOTEBOOKS

The Python Notebooks environment allows students to write code in Python, one of the world's most popular text-based programming languages. The Python Notebooks environment is made up like a notebook, with 1 or more pages shown along the top bar. Students can move between these by clicking on the page headings or by clicking the arrows at the bottom of each page. Within each page, there are information (text) boxes and code boxes. Students can edit the code boxes with their own Python code and then run this code with the Run button.



To allow students to directly interact with the Minecraft world, Python Notebooks includes a Python library (a collection of prebuilt commands) that is accessible to students. For an in depth explanation of each command available in this library, a [reference guide is available](#).

The Python Notebooks environment is best suited to students that have already an in depth understanding of coding in blocks and are looking to go a step further, or students that already have some Python coding experience outside of Minecraft Education Edition.

LESSON ACTIVITIES

Direct Instruction (Teacher-Led; “I Do”)

Recap with students on movement/controls. Ask the students:

1. Q. What were 4 keys on the keyboard that move you around in Minecraft?
A. WASD
2. Q. How can students access the Code Builder window?
A. Press the C key on the keyboard.
3. Q. How can students close the Code Builder window?
A. Press the ESC key on the keyboard.

MULTIPLAYER GROUND RULES

With this world being a multiplayer world, with potentially the entire class playing together, it is important to set some ground rules for the students. Some suggestions can be found below.

- **Do not interfere with anyone else’s work or activity**
As all students are playing together, it is possible for students to try to get in the way of other students or interfere with their work. This makes the process less enjoyable for everyone and wastes everyone’s time.
- **Engage in the tasks available**
With a world this large, it is common for students to hang around in the courtyard, the town hall, or down in the mines, avoiding completing the tasks. This once again wastes the students’ time. They are being given the freedom in this world to explore and make their own adventure, however, if they are unable to stay focused on the task, then they will no longer be able to take part in the activity.
- **If a student needs help, ask their partner first**
This helps reduce the workload on the teacher and encourages students to collaborate with their “buddy” or partner, with that student operating as a first level triage for issues or problems. Only if they are both unable to solve a particular issue, then they should come to the teacher.
- **Something not working? Double-check the instructions**
In many cases, students have struggled with some of the more complex (red level) tasks in the world simply due to then not reading the instructions for the task. Encourage students to read the instructions and plan their coding solution, before sprinting straight into coding.



Guided Instruction (Teacher-Guided; “We Do”)

Explanation

Tell the students they will be “heading on an epic adventure across the seas”, but it isn’t all going to be plain sailing... They will have to work together with their partner/buddy to solve a series of tasks to help out the residents of Seymour Island.

These tasks will be labelled with the following colours to signify difficulty

- Green – Easy/Beginner activities
- Orange – Medium/Intermediate activities
- Red – Hard/Advanced activities

Although once students complete the introduction, they can choose their own adventure pathway through the tasks. Make the strong suggestion that they should start with the beginner activities (green), before working their way up to orange and finally red.

BEGINNING GAME PLAY

As students join into the world, they will find themselves in the Lobby.

The Lobby contains several jumping challenges to keep the students occupied until it is time to start the activity.



To set up the educator controls type **/function teacher** within chat. This will give you a phone with all the commands that you need.

When you are ready to start off the activity, right-click with the phone and click on the start button. This will allocate each student currently in the game, out to an individual boat. If any students arrive late, it is possible to press the button again and they will also be allocated out onto their own individual boat.

TEACHER CONTROL BOX

Directly above the Lobby is the teacher control box. This includes a button to start the game, along with a button to teleport the teacher to the main control room (in the town).



The teacher can simply fly up, by changing their gamemode (using the `/gamemode c` command in the game chat) or by teleporting there using the following command in game chat `/tp @p -871 153 379`



When you are ready to start off the activity, right click on the button below the activity start sign (on the right in the above image). This will allocate each student currently in the game, out to an individual boat.

If any students arrive late, it is possible to press the button again and they will also be allocated out onto their own individual boat.

Task 0 is a 3-step process in which students are introduced to the basics of the Agent. This takes place for each student on their own boat, guided by Captain Papert.

TASK 0.1 LOCATE AND MOVE THE AGENT

In task 0.1, the student must locate their agent, then move it 2 blocks to the left.

If students have any issues, they should speak to Captain Papert to reset the task.

TASK 0.2 BREAKING A BLOCK

Students will have to use the destroy forward block to break the orange block in front of the Agent.

TASK 0.3 REPAIRING THE HULL

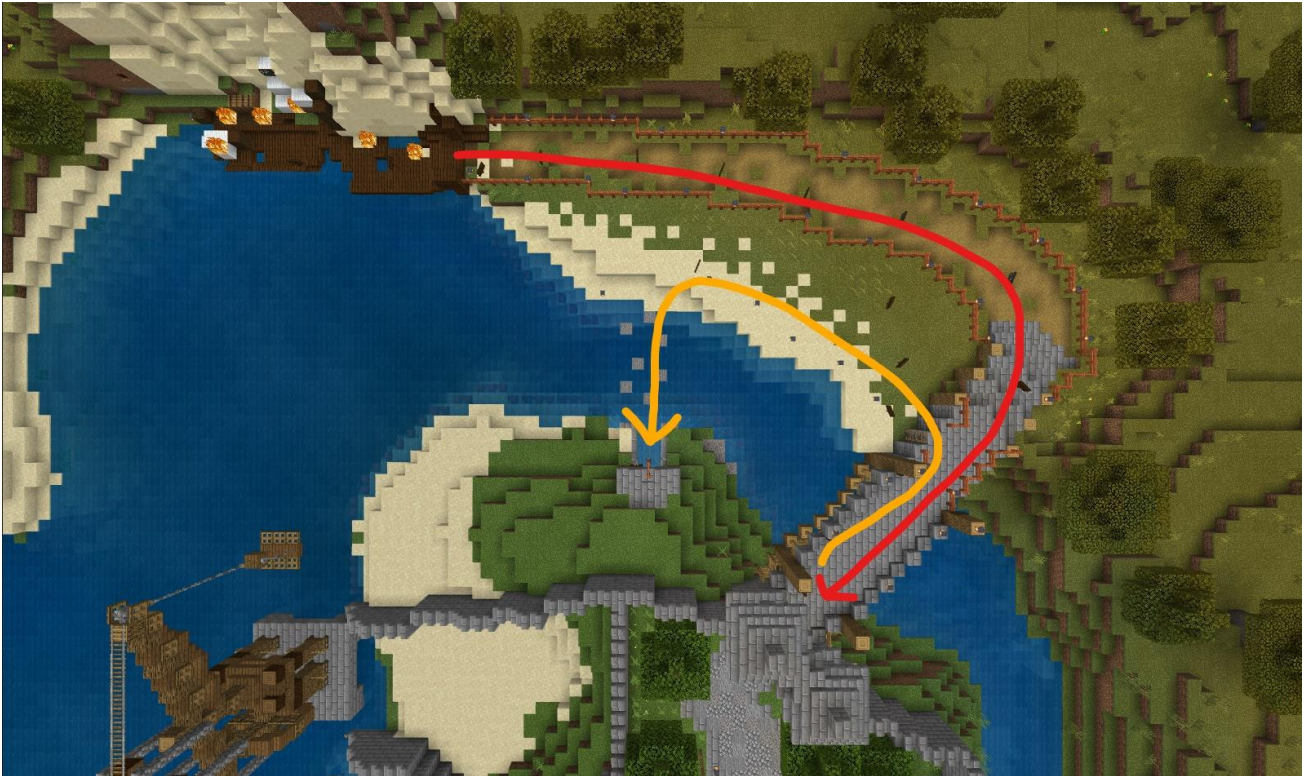
Students will have to move the agent to the left by 3 blocks, then place a block to the agents left. This will repair the hull and the player is informed that the task is complete. They will then need to speak to the captain to move on.

Once the students have spoken to the captain, they get asked to take the wheel and end up shipwrecked.



Independent Work (Teacher Support; “You Do”)

The students now need to make their way into town. To do this, they will start “blind”. Encourage students to look around to find the signs pointing them in the right direction. These signs will take them to the guard at the gate to Seymour Island, who will not let them in but mention something about a sewer. From here, students must backtrack and walk along the waterfront to the sewer. See the diagram below for the path they must follow.



Once in the sewers, students will eventually find themselves in the town square fountain. Once the students are in the town square, they can do tasks 1-8 in any order they like.

Remind the students that the tasks are colour coordinated depending on difficulty.

- Green – Easy/Beginner activities.
- Orange – Medium/Intermediate activities.
- Red – Hard/Advanced activities.

You may want to advise the students to start with the green tasks.

Task 1 – Fish and Melon

The Fish and Mellon is a beginner task located in the main town square. It involves students using their agent to place down 3 pieces of redstone to complete a circuit.

This activity is generally completed by students in 3 sections, as opposed to a full large simple program.

TASK 2 – Autominer Level 1

The autominer tasks are a series of 3 tasks that offer a progression pathway up the levels.

In Autominer 1, students are asked to dig a 7x1x3 long tunnel using the Agent. This area is behind glass which students are unable to break, so it must be done by the agent.



This task encourages students to use a Loop (For Loops) approach but can be completed by a series of smaller programs.

TASK 3 – Autominer Level 2

In Autominer 2, the task progresses from the previous with the students being required to this time, only mine the Iron Ore. Mining any other block will result in the task not being able to be completed. This requires the students to use an If Statement, as well as Loop (For Loops). This can be completed instead though with a series of smaller programs.

TASK 4 – Autominer Level 3

Autominer 3 adds 2 additional points of complexity beyond Autominer 2. It adds the requirement for students to mine any of the different ores in the wall. This requires them using If Statements and Else Statements. In addition, there is also a 50 second timer applied for this task once the agent moves off the starting position.

If the program does not successfully mine all the required ores within 50 seconds, the task (although not the students actual code) is reset.

TASK 5 – Fish Unloading

In the Fish unloading task, students are required to use their Agent to unload a total of 50 Clownfish and a total of 50 Pufferfish from the ship. Two collection hoppers can be found on the pier, where the fish should be brought to and placed in the correct hopper.

This task encourages the use of loops (While Loops).

TASK 6 – Shocking Discoveries

Shocking Discoveries is a multiplayer task, requiring 2 players. The task is a much larger and more complex version of Task 1. It features 2 circuit areas for each student, which must have redstone placed to complete the circuit. In this task, the 2 students have a very similar task to solve to automatically place the redstone, so they are encouraged to help each other.

This task encourages the use of If Statements and Loops (For Loops).

TASK 7 – Beginner Multiplayer Maze

The beginner multiplayer maze requires 2 players to complete. It involves one player being the “Driver” (they write the code) and the other being the “Navigator” (they help guide the Driver through the maze). The navigator can see the maze that must be solved but is unable to program an agent. On the other side, the Driver can write the code, but can’t actually see the maze. Whereas the driver can write code for their agent but are unable to see any of the invisible maze.

This task requires students to be ideally on a voice call (or right beside) their partner.

TASK 8 – Intermediate Multiplayer Maze

The Intermediate multiplayer maze requires 2 players to complete. It involves one player being the “Driver” (they write the code) and the other being the “Navigator” (they help guide the Driver through the maze). The navigator is able to see the maze that must be solved but is unable to program an agent. On the other side, the Driver can write the code, but can’t actually see the maze. Whereas the driver is able to write code for their agent but are unable to see any of the invisible maze. This maze contains a series of pistons and pressure pads, which must be triggered to allow the agent to pass.

This task requires students to be ideally on a voice call (or right beside) their partner.



LESSON CONCLUSION

Upon completion of this lesson, students should be able to answer these questions:

1. Q. Which activities did you find the most interesting? Why?
2. Q. Did you find working with your partner to be useful? What sort of approaches could you use in the next lesson to work better or more efficiently with them?

COMMON STUDENT ISSUES

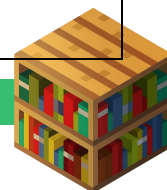
There are a number of common issues students have in Seymour Island. These can be found below:

1. **Right clicking with a blaze rod results in nothing happening, even though student code is running...**
This is a known bug within Minecraft Education Edition itself. In general, the best way to resolve it is ask the student to restart their game entirely (not just log out of the world and back in).
2. **In the autominer tasks, a student asks why they haven't completed the task, yet their agent is standing on the final pressure pad.**
This is most commonly because they either haven't actually removed all the required blocks (mainly autominer 1), or have removed too many of the stone blocks that they shouldn't have (autominer 2/3).
3. **A student's agent is missing or isn't teleporting to the right location.**
This is a known bug within Minecraft Education Edition itself. In general, the best way to resolve it is ask the student to restart their game entirely (not just log out of the world and back in).
4. **In the initial shipwreck section of the world, students are unsure of where to go after speaking with the guard.**
Encourage the students to explore, reading the clues within the world on signs. They will have to backtrack on the bridge, then climb down the ladder to the side. From there, they should be able to see a conveniently located sewer with smoke emerging.
5. **There are leftover scoreboard entries on the right. How can these be reset?**
To reset all scores for students, type the following command into the game chat
`/function reset_scores`
6. **The wrong student or multiple students have ended up inside the same task.**
This can occur due to selection limitations within Minecraft Education Edition. Encourage students to quit the task using the buttons on the wall and try and join again.
7. **The CodeBuilder opens, but all is shown is a running agent character.**
This is a known bug within Makecode within Minecraft. In general, the best way to resolve it is asking the student to restart their game entirely (not just log out of the world and back in).

KEY TEACHER COMMANDS

Minecraft standard commands

In game command	Purpose
<code>/tp STUDENT-NAME</code>	Teleport to student
<code>/tp STUDENT-NAME YOUR-NAME</code>	Teleport student to you
<code>/clear @a</code>	Clear all student inventories
<code>/give STUDENT-NAME blaze_rod</code>	Give a student a blaze rod (if they lose theirs)



/gamemode c	Set yourself to creative mode (so can fly)
-------------	--

Seymour Island specific commands

In game command	Purpose
/function lobby	Teleport to the student lobby
/function control	Teleport to the main control room
/function settings	Enable all default game settings
/function reset_scores	Reset all student scores

MAPS

Student Map



World Overview Map



SOLUTIONS

TASK 1:

MAKECODE:



PYTHON:

```
for x in range(3):  
    agent.move("forward")  
agent.move("right")  
agent.place(1, "down")
```

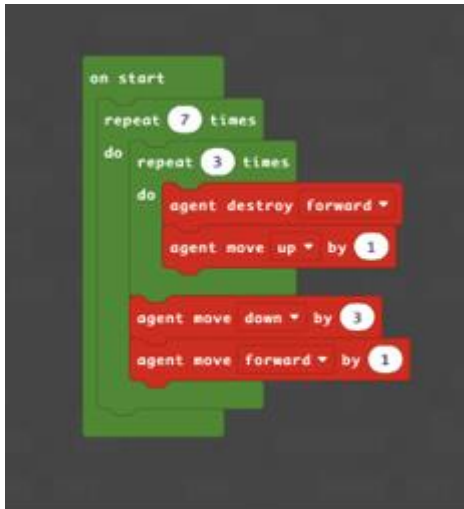
```
agent.move("right")  
agent.move("forward")  
agent.place(1, "down")
```

```
for x in range(4):  
    agent.move("left")  
agent.place(1, "down")
```



TASK 2:

MAKECODE:



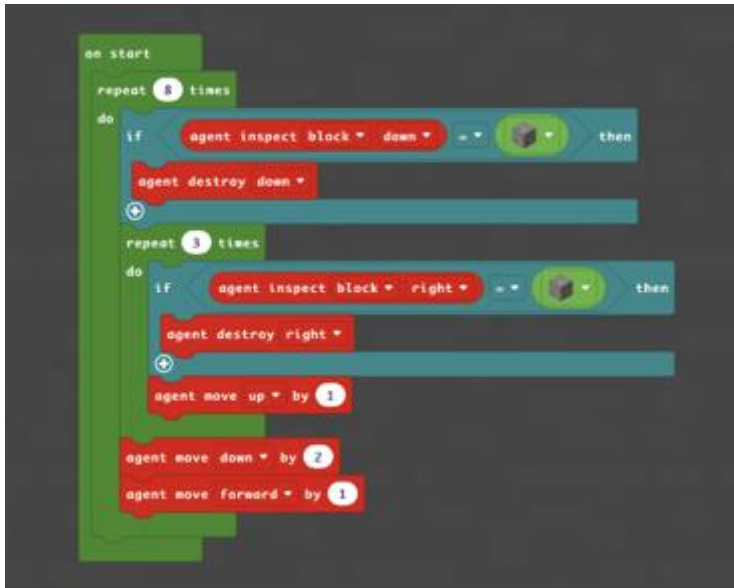
PYTHON:

```
for column in range(7):  
    agent.destroy("forward")  
  
    for height in range(2):  
        agent.move("up")  
        agent.destroy("forward")  
  
    for height in range(2):  
        agent.move("down")  
  
    agent.move("forward")
```



TASK 3:

MAKECODE:



PYTHON:

```
for row in range(12):
    if(agent.inspect("down") == "iron_ore"):
        agent.destroy("down")

    if(agent.inspect("left") == "iron_ore"):
        agent.destroy("left")

    for block_height in range(2):
        agent.move("up")
        if(agent.inspect("left") == "iron_ore"):
            agent.destroy("left")

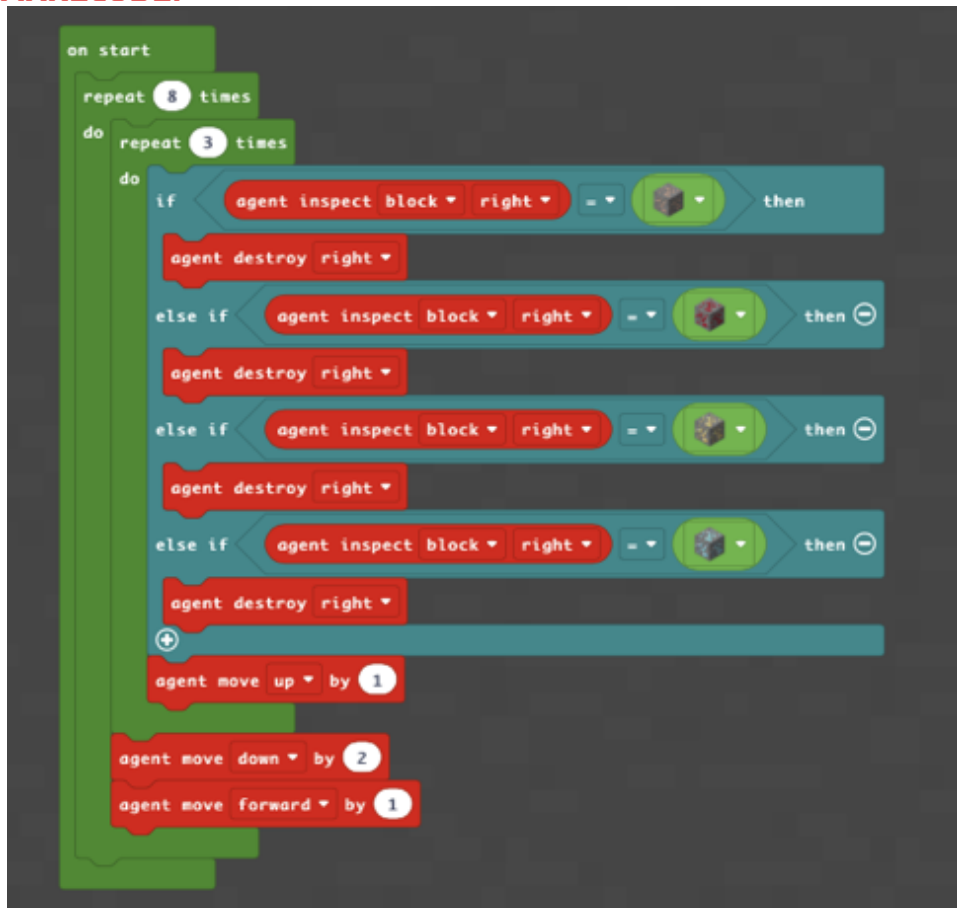
    for block_height in range(2):
        agent.move("down")

    agent.move("forward")
```



TASK 4:

MAKECODE:



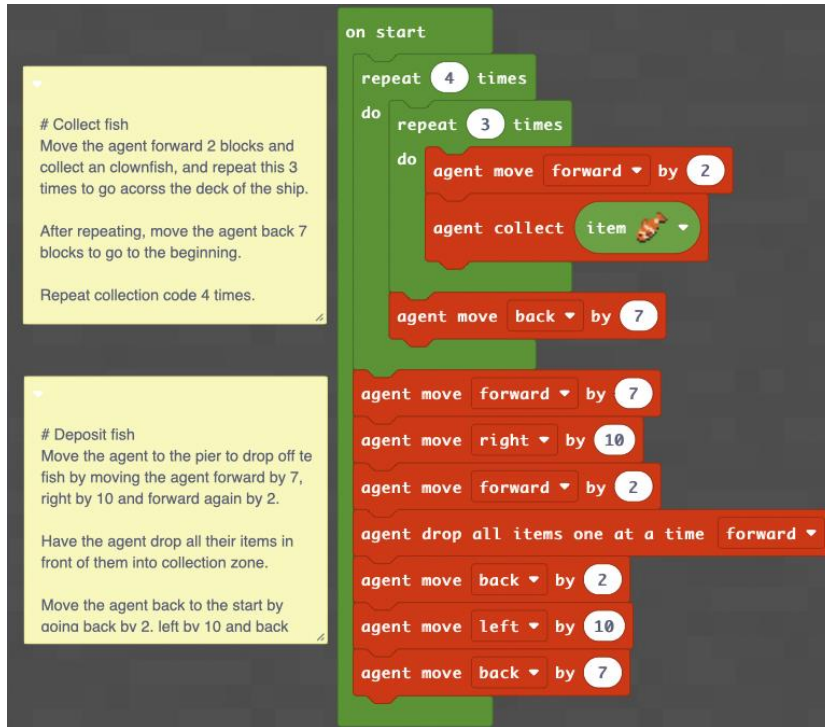
PYTHON:

```
for length in range(8):
    for height in range(3):
        # If the agent find a block that's in this list, then destroy the block.
        if agent.inspect("right") in ["iron_ore", "redstone_ore", "gold_ore", "diamond_ore"]:
            agent.destroy("right")
        agent.move("up")
    for height in range(2):
        agent.move("down")
    agent.move("forward")
```



TASK 5:

MAKECODE:



PYTHON:

```
while agent.get_item_count(0) < 50:
    for count in range(3):
        agent.move("forward")
        agent.move("forward")
        agent.collect("pufferfish")
    for count in range(0, 6):
        agent.move("back")
    agent.say("Got enough fish!")
```

```
for count in range(7):
    agent.move("forward")
for count in range(12):
    agent.move("right")
    agent.move("forward")
    agent.move("forward")
    agent.move("forward")
    agent.drop(1, 50, "down")
```

Reset agent location Using phone

```
while agent.get_item_count(0) < 50:
    for count in range(3):
        agent.move("forward")
        agent.move("forward")
        agent.collect("clownfish")
    for count in range(0, 6):
        agent.move("back")
    agent.say("Got enough fish!")
```

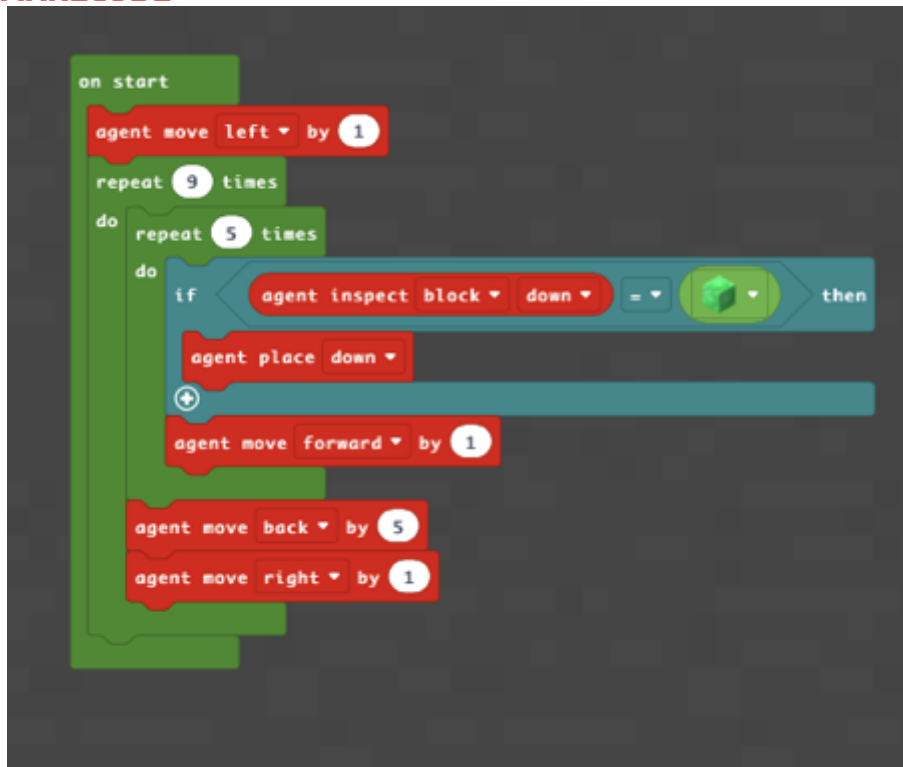


```
for count in range(7):  
    agent.move("forward")  
for count in range(10):  
    agent.move("right")  
agent.move("forward")  
agent.move("forward")  
agent.move("forward")  
agent.drop(1, 50, "down")
```



TASK 6:

MAKECODE



PYTHON:

```
agent.move("left")
```

```
for x in range(9):  
    for y in range(5):  
        if agent.inspect("down") == "emerald_block":  
            agent.place(1, "down")  
        agent.move("forward")
```

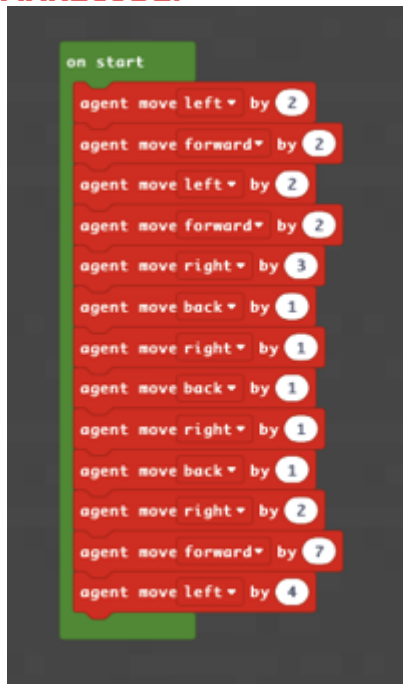
```
for y in range(5):  
    agent.move("back")
```

```
agent.move("right")
```



TASK 7:

MAKECODE:



PYTHON:

```
for x in range(2):
    agent.move("left")
for x in range(2):
    agent.move("forward")
for x in range(2):
    agent.move("left")
for x in range(2):
    agent.move("forward")
for x in range(3):
    agent.move("right")
agent.move("back")
agent.move("right")
agent.move("back")
for x in range(2):
    agent.move("right")
for x in range(7):
    agent.move("forward")
for x in range(4):
    agent.move("left")
```



TASK 8:

MAKECODE:



PYTHON:

```
for x in range(4):
    agent.move("left")
for x in range(2):
    agent.move("forward")
for x in range(2):
    agent.move("right")
for x in range(2):
    agent.move("left")
for x in range(2):
    agent.move("back")
for x in range(8):
    agent.move("right")
for x in range(6):
    agent.move("forward")
for x in range(2):
    agent.move("left")
for x in range(2):
    agent.move("right")
for x in range(6):
    agent.move("back")
for x in range(4):
    agent.move("left")
```



```
for x in range(5):
    agent.move("forward")
agent.move("left")
agent.move("forward")
agent.move("back")
agent.move("right")
for x in range(5):
    agent.move("back")
for x in range(4):
    agent.move("left")
for x in range(3):
    agent.move("forward")
agent.move("right")
for x in range(7):
    agent.move("forward")
for x in range(4):
    agent.move("right")
agent.move("back")
for x in range(3):
    agent.move("right")
for x in range(3):
    agent.move("forward")
for x in range(3):
    agent.move("left")
```



EDUCATIONAL STANDARDS

UNITED STATES	AUSTRALIA	UNITED KINGDOM
DESIGN AND ITERATIVELY DEVELOP PROGRAMS THAT COMBINE CONTROL STRUCTURES, INCLUDING NESTED LOOPS AND COMPOUND CONDITIONALS. CSTA 2-AP-12	IMPLEMENT AND MODIFY PROGRAMS WITH USER INTERFACES INVOLVING BRANCHING, ITERATION AND FUNCTIONS IN A GENERAL-PURPOSE PROGRAMMING LANGUAGE YEAR 7 AND 8 ACTDIP030	DESIGN, USE AND EVALUATE COMPUTATIONAL ABSTRACTIONS THAT MODEL THE STATE AND BEHAVIOUR OF REAL-WORLD PROBLEMS AND PHYSICAL SYSTEMS COMPUTING KEY STAGE 3
STUDENTS BREAK DOWN PROBLEMS INTO COMPONENT PARTS, EXTRACT KEY INFORMATION, AND DEVELOP DESCRIPTIVE MODELS TO UNDERSTAND COMPLEX SYSTEMS OR FACILITATE PROBLEM-SOLVING. ISTE 1.6.C	DESIGN ALGORITHMS REPRESENTED DIAGRAMMATICALLY AND IN ENGLISH, AND TRACE ALGORITHMS TO PREDICT OUTPUT FOR A GIVEN INPUT AND TO IDENTIFY ERRORS YEAR 7 AND 8 ACTDIP032	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; DESIGN AND DEVELOP MODULAR PROGRAMS THAT USE PROCEDURES OR FUNCTIONS COMPUTING KEY STAGE 3
STUDENTS UNDERSTAND HOW AUTOMATION WORKS AND USE ALGORITHMIC THINKING TO DEVELOP A SEQUENCE OF STEPS TO CREATE AND TEST AUTOMATED SOLUTIONS. ISTE 1.7.C		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. COMPUTING KEY STAGE 3



VOCABULARY GUIDE

The language in Minecraft might be new to you—here is a simple reference to terms you might come across.

Agent

A non-playable character that is a player's helper when using computer coding in-game. The agent will follow the commands input by the student and carry them out in-game. For instance, a student can have their agent build an entire structure or farm for resources.

Blocks

The basic units of structure in Minecraft. Each block measures 1 cubic meter for solids or 1000 litres for liquids.

Cryptography

The method of protecting information and secrets through the use of codes.

Python

A popular high-level text-based programming language.

Game Modes

Creative:

The creative mode lets players build anything they want from an endless supply of blocks and renders them invincible from would-be enemies.

Survival:

The survival mode, by contrast, is the standard game mode wherein players must collect resources, fight mobs, and explore -- all with the goal of staying alive.

Adventure:

Adventure is a restricted mode primarily for completing learning experiences without the ability to place or destroy blocks. Players can only move and interact with levers, buttons, and other similar items.

Griefing

The act of irritating and angering people in video games. This is done through constructing offensive builds, destroying other players builds, bullying in game, etc.

Multiplayer Experience

More than one student interacting in the same world together.

Single Player Experience

A student interacting alone within one world.

Skin

A skin is an image that determines what your character looks like in the game.



Tick

The measurement of time found in Minecraft. Time moves much faster in-game than in the real world. For example, one "day" in Minecraft lasts 20 minutes in real-time.

World

Randomly generated combinations of biomes comprise a world file for the student to work in. Worlds can be infinitely flat or mimic terrain found in biomes such as jungle, tundra, or desert, etc.

Programming

Instructions written for computers in human-readable language

Debugging

The methodical process of finding and fixing bugs that cause scripts not to do what is expected

Decomposition

Ideas, problems, or projects are broken down into component parts to set the stage for deeper analysis

