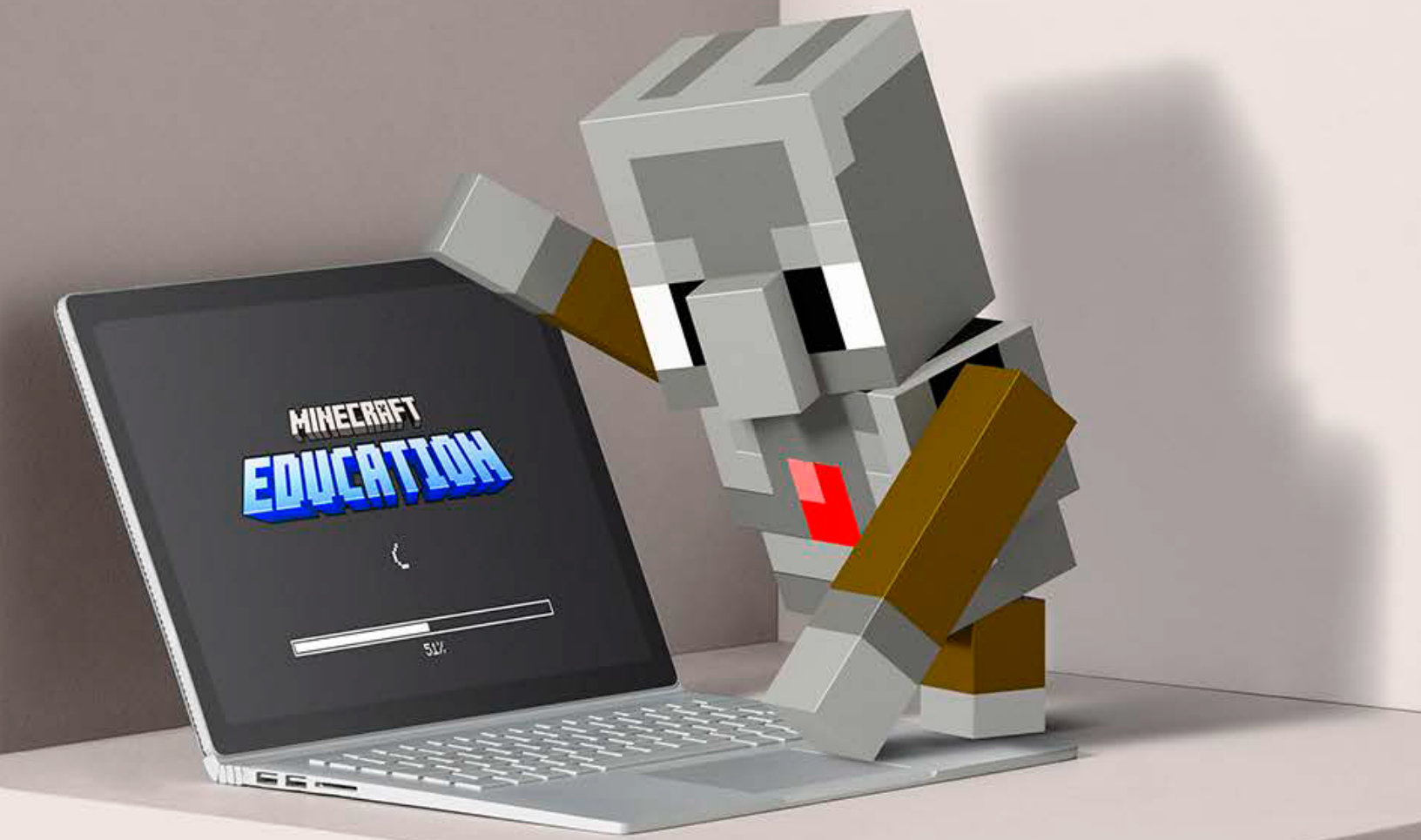
Minecraft Education & MakeCode Python

A Student Exercise Book



Introduction

Overview: Introduction to the course, its goals, and how it integrates Minecraft Education with Python programming.

Getting Started:

Setting up Minecraft Education.

Step 1: Download Minecraft Education

- Navigate to the official Minecraft Education website.
- Click on the "Download" button.
- Select the appropriate version for your operating system (Windows, Mac, iPad, or Chromebook).
- Click "Download" to start the downloading process.

Step 2: Install Minecraft Education

For Windows:

1. Once the download is complete, locate the downloaded file (usually in your "Downloads" folder).
2. Double-click the file to open the installer.
3. Follow the on-screen instructions to complete the installation.

For Mac:

1. Once the download is complete, locate the downloaded file.
2. Double-click the file to open the installer.
3. Drag the Minecraft Education icon into the "Applications" folder.

For iPad:

1. Open the App Store on your iPad.
2. Search for "Minecraft Education."
3. Tap the "Get" button and authenticate with your Apple ID if required.

For Chromebook:

1. Open the Google Play Store on your Chromebook.
2. Search for "Minecraft Education."
3. Click "Install" to download and install the app.

Step 3: Start Minecraft Education

1. Once installed, open the Minecraft Education application.
2. Log in with your school or organization account.
3. Follow the on-screen prompts to complete any initial setup.
4. Once logged in, you can start exploring the educational worlds of Minecraft or begin a new project by clicking on "Play."

Introduction to MakeCode for Python.

This course aims to bridge the captivating world of Minecraft with the power of Python programming. You will learn to harness the creative potential of Minecraft Education Edition while gaining a solid foundation in coding through MakeCode.

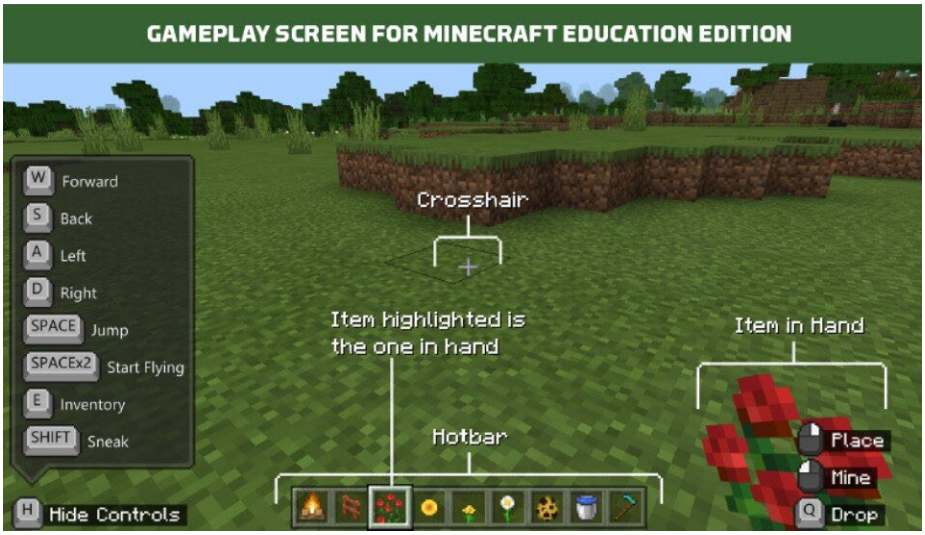
By the end of the course, students will have developed the skills to create their own scripts, manipulate the Minecraft environment, and understand fundamental programming concepts.

Navigating the Minecraft environment.

Navigating the Minecraft environment within the Education Edition involves understanding the basic controls, interface, and key features that will enhance your learning experience.

First, familiarize yourself with the movement controls: use the **W, A, S, and D keys** to move forward, left, backward, and right, respectively. The space bar allows you to jump, while double-tapping the space bar enables flying mode, which can be very useful for surveying your surroundings.

The **mouse** is used to look around and interact with the world. Left-click to break blocks or attack entities, and right-click to place blocks or use items. The inventory, accessed by pressing the E key, is where you manage your items and blocks. Here, you can drag and drop items to organize your inventory or craft new items using the crafting grid.



The **hotbar** at the bottom of the screen shows your quick access items. Use the number keys (1-9) to select an item from the hotbar. To view your current objective or access the menu, press the ESC key. This menu allows you to adjust settings, save your progress, or exit the game.

Understanding coordinates is essential for navigation and programming within Minecraft. Press F3 to bring up the debug screen, which displays your current coordinates (x, y, z). These coordinates will help you locate specific points in the world, set up teleportation scripts, and build accurately.

With these basics covered, you are ready to start coding and creating within Minecraft Education.

Basic coding concepts overview.

Python is a versatile and beginner-friendly programming language that will be used throughout this course to interact with the Minecraft environment. By leveraging Python within Minecraft Education Edition, you can create scripts that automate tasks, manipulate the game world, and develop a deeper understanding of programming fundamentals.

At the core of Python coding in Minecraft, you will start with simple commands such as `print()`, which outputs text to the console, and gradually move to more complex operations like running, pausing scripts, and connecting Minecraft to MakeCode. This structured approach ensures that you will build your skills step-by-step, from the basics to more advanced concepts.

Unit 1: Getting Started with Python in Minecraft

Lesson 1.1: Setting Up Your First Python Script

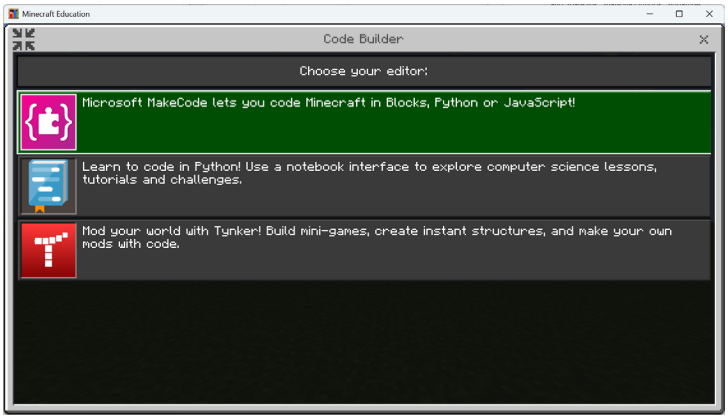
Lesson 1.1: Setting Up Your First Python Script

Welcome to your first lesson on creating a Python script in Minecraft Education using the MakeCode interface! In this lesson, you will learn the essential steps to set up your script, navigate the MakeCode environment, and execute basic commands.

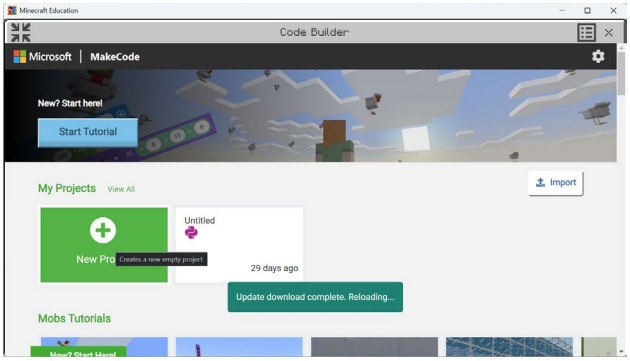
Step 1: Accessing the MakeCode Interface

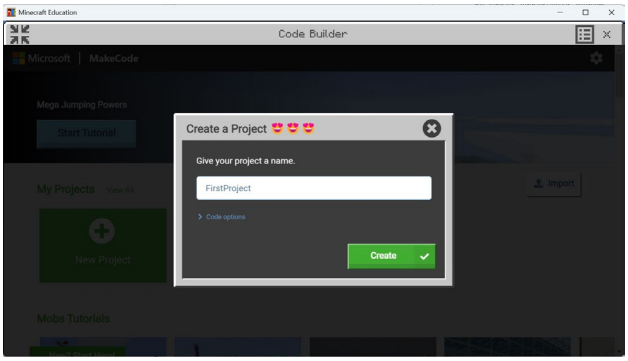
1. Launch Minecraft Education Edition on your device.
2. Create a new world or click this link https://education.minecraft.net/world/6d02471a-4d57-4357-a63f-912b37c9d9c5 to open a flat world.

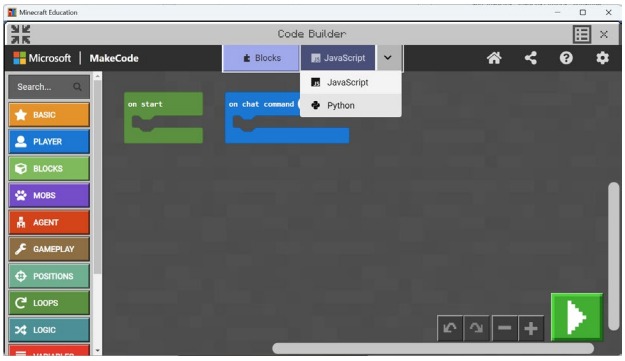
3. Press 'C' on your keyboard to open the MakeCode interface.

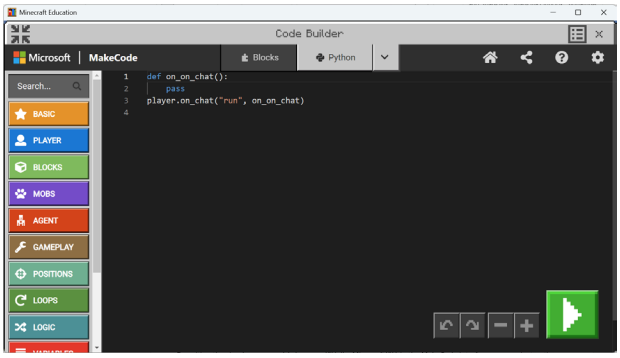


4. Select 'New Project' and name your project. Then Select the Python interface from the dropdown menu.







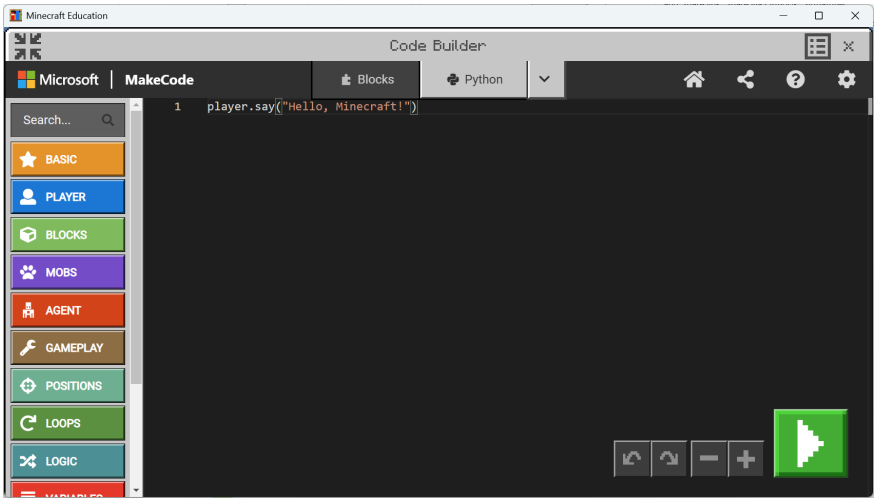


Step 2: Writing Your First Script

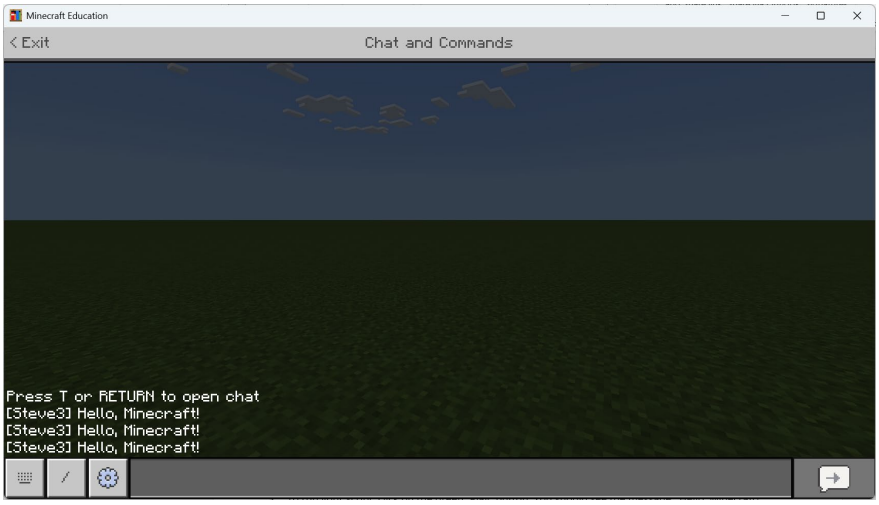
1. In the MakeCode interface, delete the code that is there as default.

2. Start by writing a simple command to display a message in the console. Type the following code:

```
player.say("Hello, Minecraft!")
```



3. To run your script, click on the green 'Play' button. You should see the message "Hello, Minecraft!" appear in the console.



Step 3: Understanding the Basics

By writing the simple command `player.say("Hello, Minecraft!")` in the MakeCode interface, you are instructing the Minecraft player entity to display the message "Hello, Minecraft!" in the message window. This command is executed when you click the green 'Play' button, which runs the script.

The `player.say()` function is specifically designed to output text messages from the player's perspective in the game environment, and it serves as a basic example of how you can use Python to interact with Minecraft. This foundational command introduces you to the concept of scripting in Minecraft Education, allowing you to see immediate results from your code.

As you progress, this basic understanding will enable you to experiment with more complex commands and scripts, ultimately enhancing your ability to create intricate and interactive Minecraft experiences.

Step 4: Saving Your Work

1. Your work is automatically saved in the MakeCode interface. You can save your world by pressing **esc** and clicking save & exit at any time.
2. Ensure your project is saved regularly to avoid losing your progress.

Step 5: Experimenting with Commands

1. Try modifying the `player.say()` function to display different messages.
2. Experiment with adding multiple `player.say()` statements to understand how the console outputs each line in sequence.

With these steps, you have successfully set up and executed your first Python script in Minecraft Education using the MakeCode interface. This foundational knowledge will be crucial as you progress through the course and learn more advanced coding techniques.

Next, you will move on to Lesson 1.2, where you will learn to navigate the Minecraft world using coordinates and write your first teleportation script.

Lesson 1.2: Understanding the Minecraft World

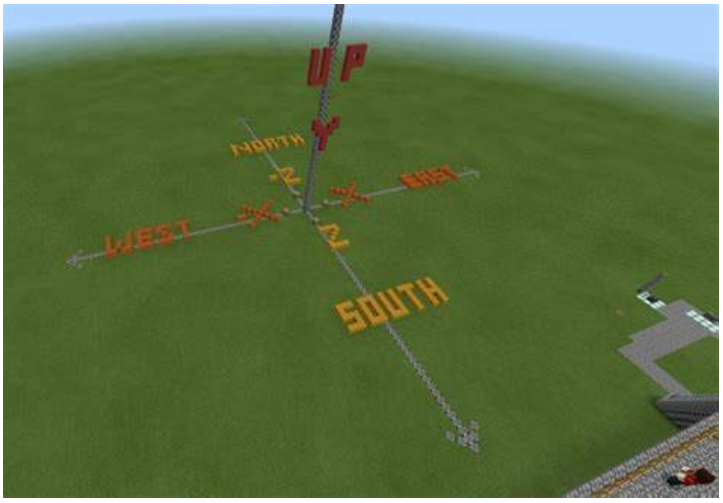
Navigating coordinates (x, y, z).

Understanding Coordinates in Minecraft Education

Coordinates in Minecraft Education are a crucial aspect of navigating and manipulating the game world. They allow you to pinpoint specific locations and execute precise commands. Coordinates are represented as three values: X, Y, and Z.

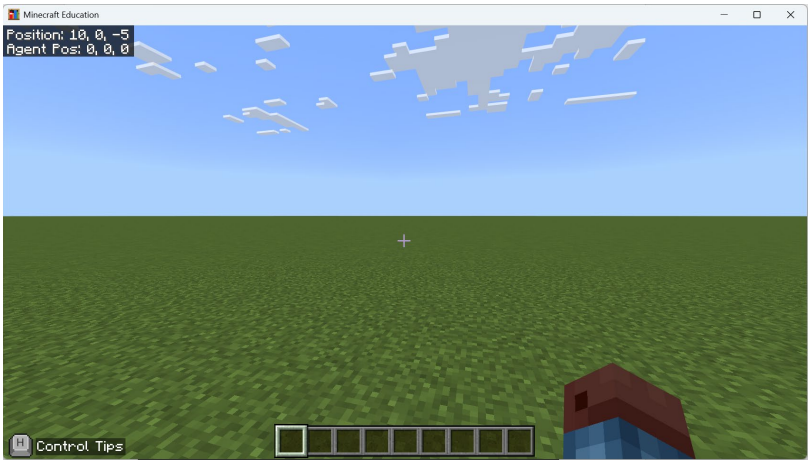
X, Y, and Z Coordinates

- X: This coordinate represents the player's position east or west of the origin point. Positive values indicate east, while negative values indicate west.
- Y: This coordinate represents the player's vertical position. It indicates how high or low you are in the world, with higher values denoting greater elevation.
- Z: This coordinate represents the player's position north or south of the origin point. Positive values indicate south, while negative values indicate north.



Navigating Using Coordinates

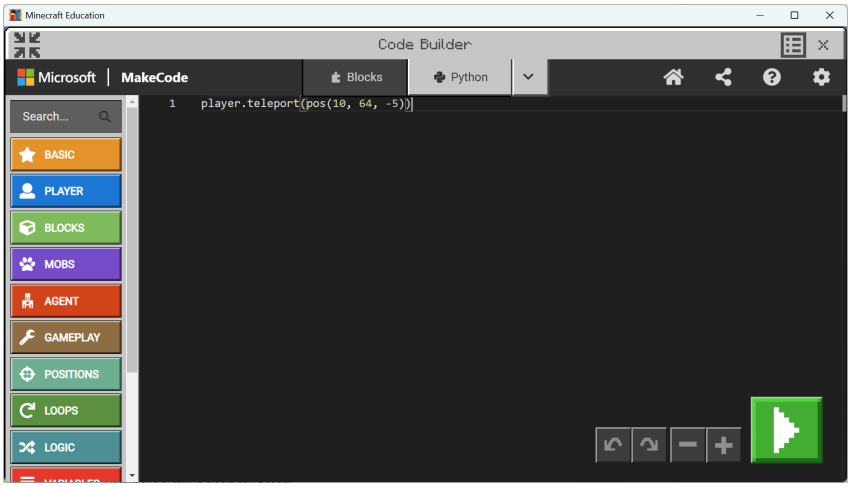
Coordinates are displayed on the screen when you press the **F1 key** (or toggle coordinates in settings on some versions). Knowing your coordinates can help you navigate the world more effectively. For instance, if you are mining and need to find your way back to a specific spot, keeping track of the coordinates will ensure you can return precisely to that location.



Writing Your First Teleportation Script

Using coordinates, you can write scripts to teleport to various locations within the game. The teleportation command in Python might look as follows:

```
player.teleport(pos(10, 64, -5))
```



This command would teleport you (the player) to the coordinates X=10, Y=64, and Z=-5. Mastering the use of coordinates will significantly enhance your ability to create intricate Minecraft scripts and navigate the game world with ease.

With these foundational skills, you will be well-prepared to explore more complex coding techniques and projects in Minecraft Education.

In this code we use a *method*. A method in programming is used to perform a specific action. In the context of the player.teleport() example, the method "teleport" belongs to the "player". When you call

player.teleport(pos(10, 64, -5)), you are instructing the player to execute its teleport function, which moves the player to the specified coordinates.

Lesson 1.3: Basic Blocks and Commands

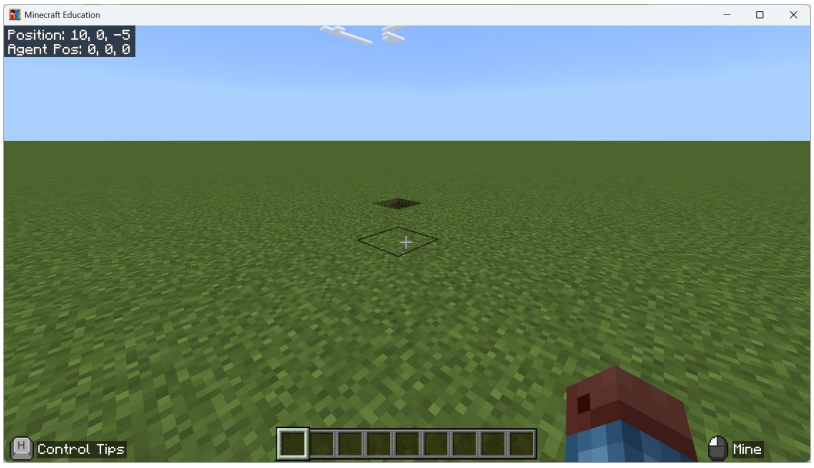
Placing and breaking blocks in Minecraft using Python code is an essential skill for creating automated structures and environments.

To place a block, you use the `blocks.place()` method. This method requires the coordinates where you want to place the block and the type of block you wish to place. For instance, the following code places a stone block at coordinates (5, 5, -5):

```
# Coordinates where the block will be placed

x, y, z = 5, 5, -5

# Place a stone block

blocks.place(STONE, positions.create(x, y, z))
```

Breaking a block involves replacing the block at a specified location with an 'air' block, essentially removing the existing block. Here is an example:

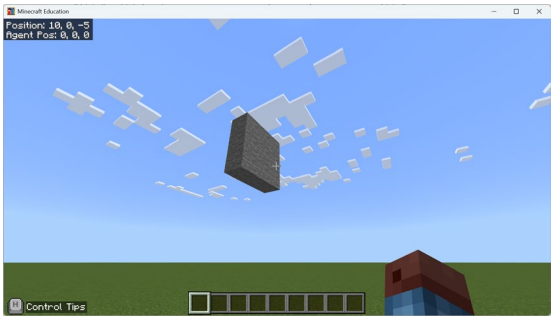
```
# Coordinates of the block to be broken

x, y, z = 5, -1, -5


# Break the block by placing air

blocks.place(AIR, positions.create(x, y, z))
```



This created a hole of ‘AIR’ at 5, -1, 5, which is in the ground on which I stand.

By mastering these commands, you can manipulate the Minecraft world programmatically, creating intricate designs and automating tasks with ease. As you progress, these basic operations will be the foundation for more complex projects, such as constructing entire buildings or landscapes dynamically within the game.

Create a STONE wall



Creating a STONE wall in Minecraft using code involves placing multiple blocks in a structured manner. The provided code demonstrates how to manually place each block to form a 3x3 STONE wall.

To start, you need to define the starting coordinates of your wall. In this example, the wall begins at coordinates (5, 5, -5). From there, the blocks are placed row by row. First, the code places the blocks for the first row at (5, 5, -5), (6, 5, -5), and (7, 5, -5). Next, the blocks for the second row are placed directly above the first row at (5, 6, -5), (6, 6, -5), and (7, 6, -5). Finally, the third row is placed above the second row at (5, 7, -5), (6, 7, -5), and (7, 7, -5). By following this pattern, you can extend the wall to any size by continuing to add coordinates in the desired rows and columns.

```
# Starting coordinates for the wall
x_start, y_start, z_start = 5, 5, -5


# Manually place each block in the wall (3x3 example)
blocks.place(STONE, positions.create(x_start, y_start, z_start))      # Row 1, Column 1
blocks.place(STONE, positions.create(x_start + 1, y_start, z_start))  # Row 1, Column 2
blocks.place(STONE, positions.create(x_start + 2, y_start, z_start))  # Row 1, Column 3


blocks.place(STONE, positions.create(x_start, y_start + 1, z_start))  # Row 2, Column 1
blocks.place(STONE, positions.create(x_start + 1, y_start + 1, z_start))  # Row 2, Column 2
blocks.place(STONE, positions.create(x_start + 2, y_start + 1, z_start))  # Row 2, Column 3


blocks.place(STONE, positions.create(x_start, y_start + 2, z_start))  # Row 3, Column 1
blocks.place(STONE, positions.create(x_start + 1, y_start + 2, z_start))  # Row 3, Column 2
blocks.place(STONE, positions.create(x_start + 2, y_start + 2, z_start))  # Row 3, Column 3
```