Python and Minecraft

Contents

Minecraft Intro

- Looking and moving around
- Building and destroying blocks
- Hello world with Python
- Moving the player with Python
- Creating blocks with Python

Getting started with Minecraft

Double click the `Minecraft Pi Edition` icon on the desktop.

When Minecraft Pi has loaded, click on Start Game, followed by Create new. You'll notice that the containing window is offset slightly. This means to drag the window around you have to grab the title bar behind the Minecraft window.

Leave the window the size it is, if it's maximised it can cause problems later.

Use the mouse to look around, and the following keys:

Key	Action
W	Forward
A	Left
S	Backward
D	Right
E	Inventory
Space	Jump
Double Space	Fly / Fall
Esc	Pause / Game menu
Tab	Release mouse cursor

You can select an item from the quick draw panel with the mouse's scroll wheel (or use the numbers on your keyboard), or press E and select something from the inventory.

You can also double tap the space bar to fly into the air. You'll stop flying when you release the space bar, and if you double tap it again you'll fall back to the ground.

With the sword in your hand, you can click on blocks in front of you to remove them (or to dig). With a block in your hand, you can use right click to place that block in front of you, or left click to remove a block.

Hello World with Python

With Minecraft running, bring the focus away from the game by pressing the Tab key, which will free your mouse. Open Python 3 by double clicking the `IDLE 3` desktop icon.

The first program you'll write will print a message on the Minecraft screen.

In the Python Idle window, click `File > New window`. This will open a new window which you're going to write the program in.

Type the following in, paying attention to capitals and punctuation!

```
import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

mc.postToChat('Hello World!')
```

The first line uses the `import` command to load the Minecraft library. The second line creates the interface between Python and Minecraft, and the final `postToChat` command is what prints the message.

Save the file as `message.py` by clicking `File > Save`. You can then run the program by pressing the F5 key on the keyboard.

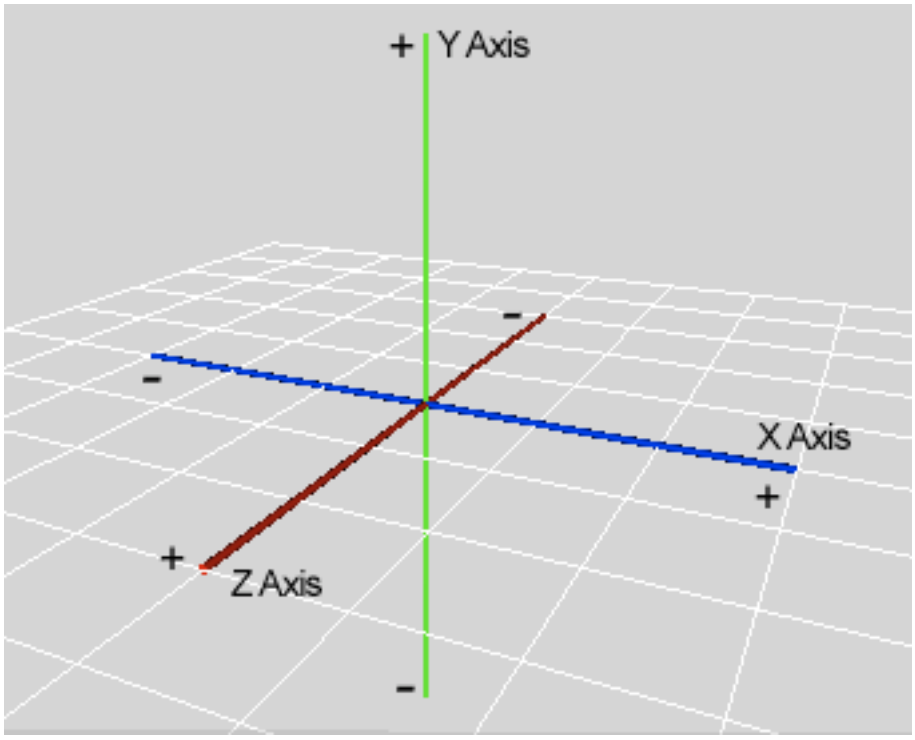
Find your location

For each program you write, we recommend you create a new file by clicking `File > New window`. Feel free to copy and paste the parts you need from your previous programs.

This program starts exactly the same way, but then stores your location into 3 variables and later prints them out.

```
import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

x, y, z = mc.player.getPos()
print(x, y, z)
```

The x, y, and z variables contain each part of your position coordinates: x and z are the walking directions (forward/back and left/right) and y is up/down.



Try running the program with F5 and check that the location matches the numbers printed in the top left of the screen. Then move somewhere else and run the program again - do the numbers still match?

Teleport

As well as finding out your current location you can specify a particular location to teleport to. Start a new program and the save it as `teleport.py`.

```
import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

x, y, z = mc.player.getPos()
mc.player.setPos(x, y+100, z)
```

This program will first get your location and store it in the `x`, `y` and `z` variables. Then it uses `setPos` to transport your player 100 spaces higher up than you currently are. This means you'll teleport to the middle of the sky and fall straight back down to where you started.

Try teleporting to somewhere else!

Single blocks

To place a block, use the `setBlock` code:

```
mc.setBlock(x, y, z, block_id)
```

`x`, `y` and `z` set the position of the block, and `block_id` sets the type of block.

This program should set a block right next to your player:

```
import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()

x, y, z = mc.player.getPos()
mc.setBlock(x + 1, y, z, 41)
```

If you can't see the block, it might be because it's behind you! Try turning around with the mouse.

Each block has a number and a name. For example, the block you created with the last program was a gold block. Its number is 41 and the name is `GOLD_BLOCK`.

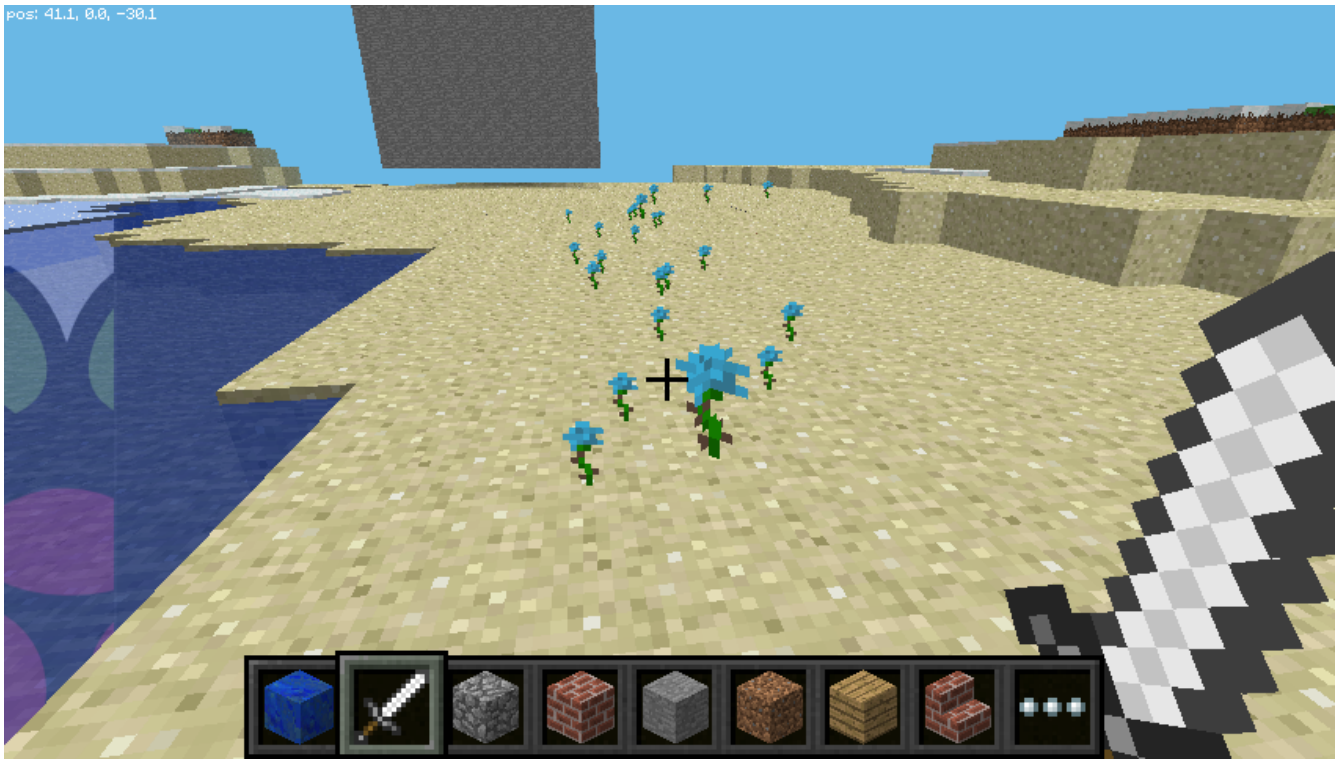
Sometimes it's clearer to create a block with the name instead of the number. We can do that by importing another library first:

```
import mcpi.minecraft as minecraft
import mcpi.block as block

mc = minecraft.Minecraft.create()

x, y, z = mc.player.getPos()
mc.setBlocks(x + 1, y, z, block.GOLD_BLOCK.id)
```

Here are a list of all the blocks:


AIR 0
STONE 1
GRASS 2
DIRT 3
COBBLESTONE 4
WOOD_PLANKS 5
SAPLING 6
BEDROCK 7
WATER 9
LAVA 11
SAND 12
GRAVEL 13
GOLD_ORE 14
IRON_ORE 15
COAL_ORE 16
WOOD 17
LEAVES 18
GLASS 20
LAPIS_LAZULI_ORE 21
LAPIS_LAZULI_BLOCK 22
SANDSTONE 24
BED 26
COBWEB 30
GRASS_TALL 31
WOOL 35
FLOWER_YELLOW 37
FLOWER_CYAN 38
MUSHROOM_BROWN 39
MUSHROOM_RED 40
GOLD_BLOCK 41
IRON_BLOCK 42
STONE_SLAB_DOUBLE 43
STONE_SLAB 44
BRICK_BLOCK 45
TNT 46
BOOKSHELF 47
MOSS_STONE 48
OBSIDIAN 49
TORCH 50
STAIRS_WOOD 53
CHEST 54
DIAMOND_ORE 56
DIAMOND_BLOCK 57
CRAFTING_TABLE 58
FARMLAND 60
FURNACE_INACTIVE 61
FURNACE_ACTIVE 62
DOOR_WOOD 64
STAIRS_COBBLESTONE 67
DOOR_IRON 71
REDSTONE_ORE 73
SNOW 78
ICE 79
SNOW_BLOCK 80
CACTUS 81
CLAY 82
SUGAR_CANE 83
FENCE 85
GLOWSTONE_BLOCK 89
BEDROCK_INVISIBLE 95
STONE_BRICK 98
GLASS_PANE 102
MELON 103
FENCE_GATE 107
GLOWING_OBSIDIAN 246
NETHER_REACTOR_CORE 247

Creating blocks as you walk



Now you know how find your location using `getPos` and create a block using `setBlock`, let's put it together in a loop, so that wherever you walk you create a trail of blocks behind you automatically.

```
import mcpi.minecraft as minecraft
import mcpi.block as block

mc = minecraft.Minecraft.create()

while True:
    x, y, z = mc.player.getPos()
    mc.setBlocks(x, y, z, block.FLOWER_CYAN.id)
```

Since we used a while True loop this will go on forever. To stop it, hit Ctrl + C in the Python window.

Try a few different types of blocks. Can you work out how to change the code so the blocks get created above you or below you?

Traffic lights

- This activity will help you understand loops using Python.
- You will need: Minecraft with the Python API

You will create traffic lights in Minecraft and animate them with a loop.

Figure 1: traffic lights

Creating blocks

You need to import the Minecraft library and setup the `handle`:

```
import mcpi.minecraft as minecraft
import mcpi.block as block

mc = minecraft.Minecraft.create()
```

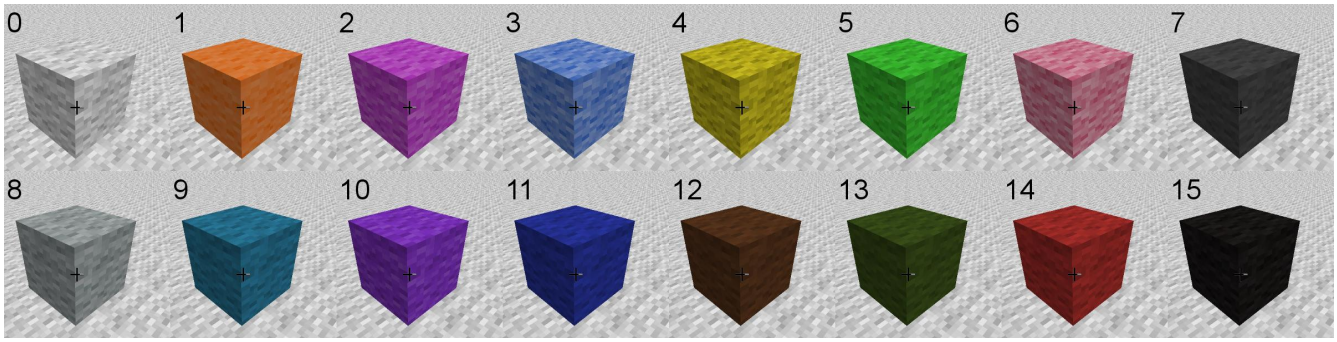
Then clear a space and send your player there:

```
# clear area
mc.setBlocks(-60,0,-60,60,50,60,block.AIR.id)

# go there
mc.player.setPos(5,0,0)
```

Either use a few `setBlock` commands or a single `setBlocks` command to create a traffic light.

Coloured wool blocks

To change colours of blocks in Minecraft, you can use the `WOOL` block. It's a special block that can be set to 16 different colours:



Here's how you could create a black `WOOL` block at the 0, 0, 0 position.

```
mc.setBlock(0, 0, 0, block.WOOL.id, 15)
```

In this case, you pass an extra parameter (15) to setBlock which sets the wool block to be black.

Loops

To create an animation of the lights changing you'll use an infinite loop. Here's an example that flashes a block between black and red:

```
import time
while True:

    # make block red
    mc.setBlock(0, 0, 0, block.WOOL.id, 14)
    # wait for 1 second
    time.sleep(1)

    # make block black
    mc.setBlock(0, 0, 0, block.WOOL.id, 15)
    # wait for 1 second
    time.sleep(1)
```

Now extend the loop to animate the whole traffic light sequence.

Teleport

- This activity will help you understand how to use a Pibrella with Minecraft
- You will need: Minecraft with the Python API
- PiBrella add on board

You will use the PiBrella's button to teleport your player to a random location in Minecraft.

Button

To check the PiBrella is working, copy this program and save it as `teleport.py`.

```
import pibrella

while True:
    if pibrella.button.read() == 1:
        print("button pressed!")
```

If you try and run the program with F5 as usual, it will fail with the error message "You must be root to use Pibrella!"

This is because the PiBrella needs super user access to work, which means you need to run the program with the `sudo` command (a shortening of super user do).

First, open a terminal by double clicking the `LXTerminal` icon on the desktop. This opens a black window where you can type the following commands:

```
sudo python teleport.py
```

When you press the button you should see the message being printed - probably lots of times. Can you press the button quickly enough to only print one message? Why do you think more than 1 messages are being printed?

You can slow the loop down by using the `sleep` command. First import the time library:

```
import time
```

Then inside the loop, add a small delay:

```
time.sleep(1)
```

Run the program again, can you get just one message now?

Random numbers

For our teleport program to work, you need to pick some random numbers to teleport to once the button is pressed.

To get a random number, you can use the `random` library. Try running this program a few times:

```
import random
x = random.randint(-100, 100)
print(x)
```

The `randint(min, max)` function takes 2 parameters that set the range of the random number it returns.

Modify the button test program above so that 3 new random numbers are printed every time the button is pressed.

Teleport

Teleporting in Minecraft is as easy as setting the player position:

```
import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()
mc.player.setPos(100, 100, 100)
```

Modify your program so that pressing the button teleports you to a random new location.

Trap!

- This activity will help you understand positions in Minecraft
- conditional statements in Python

You will build a trap, then create a program that builds a wall the instant you step inside.

Building a trap

This time, you'll build a 3 walled cubicle by hand.

- Use the mouse to look around. Left click destroys a block, right click places a block.
- Use number keys or mouse scroll wheel to choose what block to place.
- Use the inventory (press the E key) to select different block types.

Figure 2: trap

Positions

You'll need the usual lines at the start of your program:

```
import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()
```

Now add the following code to print out your location:

```
while True:
    playpos = mc.player.getTilePos()
    print(playpos)
```

Run the program and walk around. You should see your position in x, y, z co-ordinates being printed in the Python Shell.

Walk to your trap and make a note of the co-ordinates when you are inside.

Conditionals

Now you need a `conditional statement` so that something happens only when you're in the exact location you just found. Don't use my numbers below, use the numbers you found when you ran your program.

```
if playpos.x == -247 and playpos.y == 10 and playpos.z == 60:
    print("trapped!")
```

Put this code into your loop (make sure the code is indented properly) and run your program again. This time, when you walk into the trap the program should print out 'trapped!'.

Now change your program so instead of printing a message it builds a wall behind you after you've walked in - preventing you from walking out.

Photo booth

- This activity will help you understand positions in Minecraft,
- Conditional statements in Python,
- How to use a camera with Python,
- You will need: Minecraft with the Python API, USB webcam or a PiCamera.

You will build a photo booth, then create a program that takes a photo when you step inside. Its just like the trap activity, but it takes a photo instead of trapping you.

Building the booth

Use your building skills to construct a 3 walled cubicle like the picture below:

Figure 3: booth

Positions

Start your program with the usual lines:

```
import mcpi.minecraft as minecraft
import mcpi.block as block

mc = minecraft.Minecraft.create()
```

Now add the following code to print out your location:

```
while True:
    playpos = mc.player.getTilePos()
    print(playpos)
```

Run the program and walk around. You should see your position in x, y, z co-ordinates being printed in the Python Shell.

Walk to your booth and make a note of the co-ordinates when you are inside.

Conditionals

Now you need a `conditional statement` so that something happens only when you're in the exact location you just found:

```
if playpos.x == -247 and playpos.y == 10 and playpos.z == 60:
    print("say cheese!")
```

Put this code into your loop (make sure the code is indented properly) and run your program again. This time, when you walk into the booth the program should print out 'say cheese!'.

Now change your program so instead of printing a message it takes a photo of you using an attached web cam or PiCamera.

Webcam

If you have a webcam, first install the fswebcam program by opening a terminal and typing:

```
sudo apt-get install fswebcam
```

Then this Python code will take a photo:

```
filename = 'image.jpg'
import os
# os.system() runs a linux command. fswebcam is a program that can take photos
os.system("fswebcam  --no-banner -r 800x600 -d /dev/video0 " + filename)
```

Picamera

The PiCamera needs to be installed along with the PiCamera Python library.

```
filename = 'image.jpg'
import picamera
camera = picamera.PiCamera()
camera.capture(filename)
```

Big buildings

- This activity will help you understand positions in Minecraft,
- How to define volumes in 3D space
- Loops in Python,

It's possible to build amazing things in the Minecraft world, and it can often be a lot quicker with a few Python tricks!

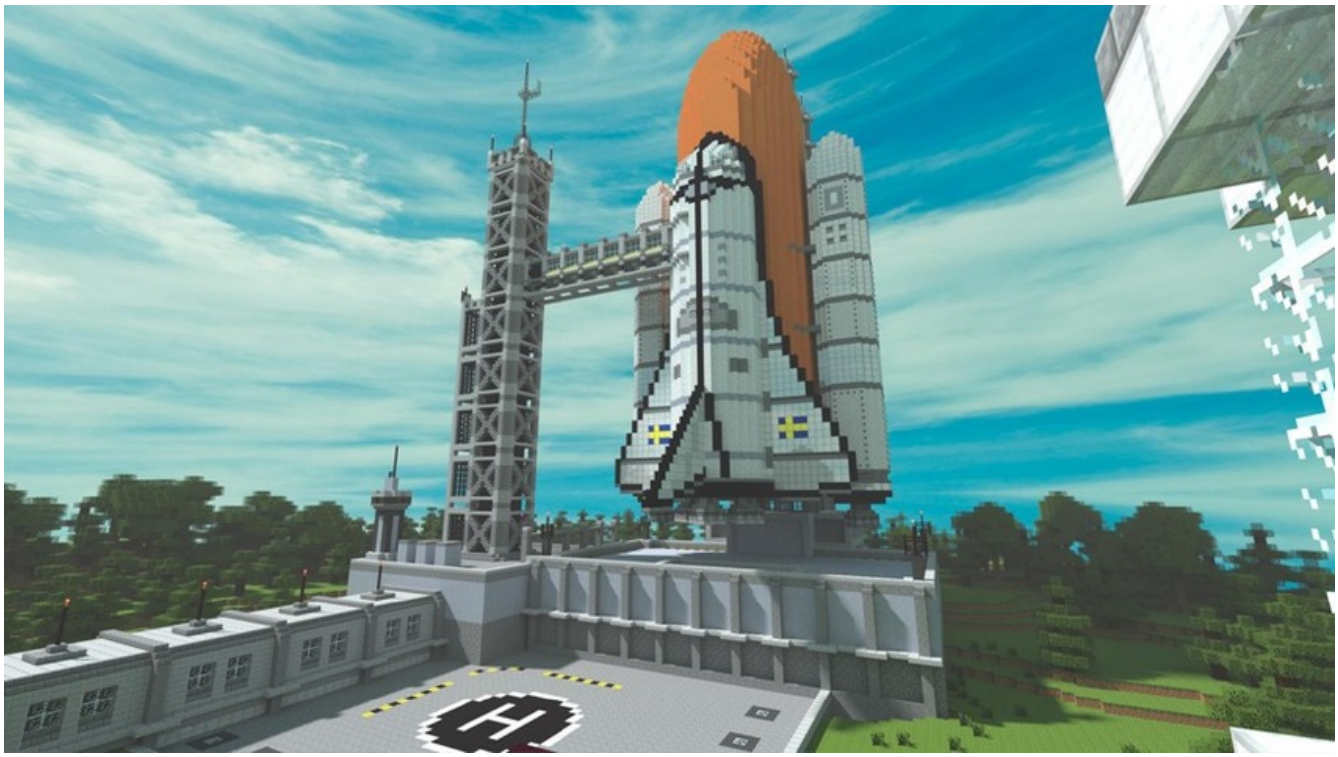
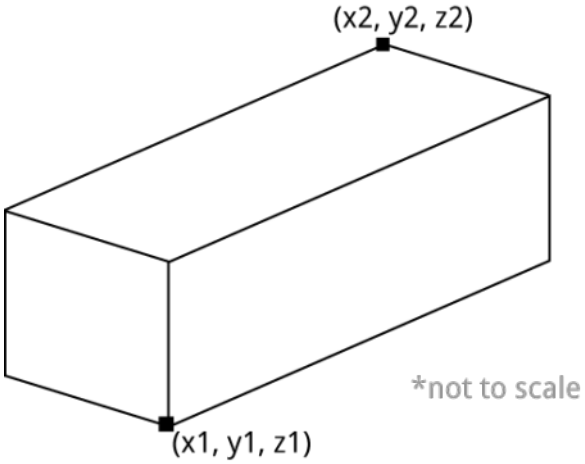


Image courtesy of crpeh

Bigger blocks

You've already seen how to build single blocks at a time with the `setBlock` function.

Placing one block at a time is great, but there's also a way to creat big volumes of blocks by asking Minecraft to fill in all the space in between 2 co-ordinates:



The volume between `x1, y1, z1` and `x2, y2, z2` will be filled with blocks of type `block_id`. Here's what the code would look like for the picture above:

```
setBlocks(x1, y1, z1, x2, y2, z2, block_id)
```

Note that `x1`, `y1`, `z1`, `x2`, `y2` and `z2` are just telling you what parameters are needed and how they work. You'd need to put real numbers in to make it work.

To use a real example, let's say you want to make a cube 20 blocks wide, with the center at x=0, y=0, z=0. The code would be:

```
setBlocks(-10, -10, -10, 10, 10, 10, blocks.GOLD_BLOCK.id)
```

An easy way to clear a big space ready to start building is to use `setBlocks` to fill a volume with air blocks:

```
mc.setBlocks(-60, 0, -60, 60, 60, 60, block.AIR.id)
```

This command would remove a large cuboid, 120 blocks deep by 120 blocks wide and 60 blocks tall.

Tower blocks

Try using `setBlocks` to build a few big blocks.

If you use a loop, you could build a tower block with each story built of a different kind of block:

```
story = 0
block_id = 10
# a 10 story building
while story < 10:
    mc.setBlocks(-5, story, -5, 5, story, 5, block_id)
    story = story + 1
    block_id = block_id + 1
```

Pyramids

Try building a pyramid by stacking 5 squares on top of each other, with each square a bit smaller than the last:

```
mc.setBlocks(-5,0,-5,5,0,5,block.GOLD_BLOCK.id)
mc.setBlocks(-4,1,-4,4,1,4,block.GOLD_BLOCK.id)
mc.setBlocks(-3,2,-3,3,2,3,block.GOLD_BLOCK.id)
...
```

Complete the pattern and test your code, does it make a pyramid?

Now you'll use a loop to build a gigantic pyramid without having to do lots of repetitive typing.

There are 3 patterns in the numbers. Remember the definition of the `setBlocks` commmand:

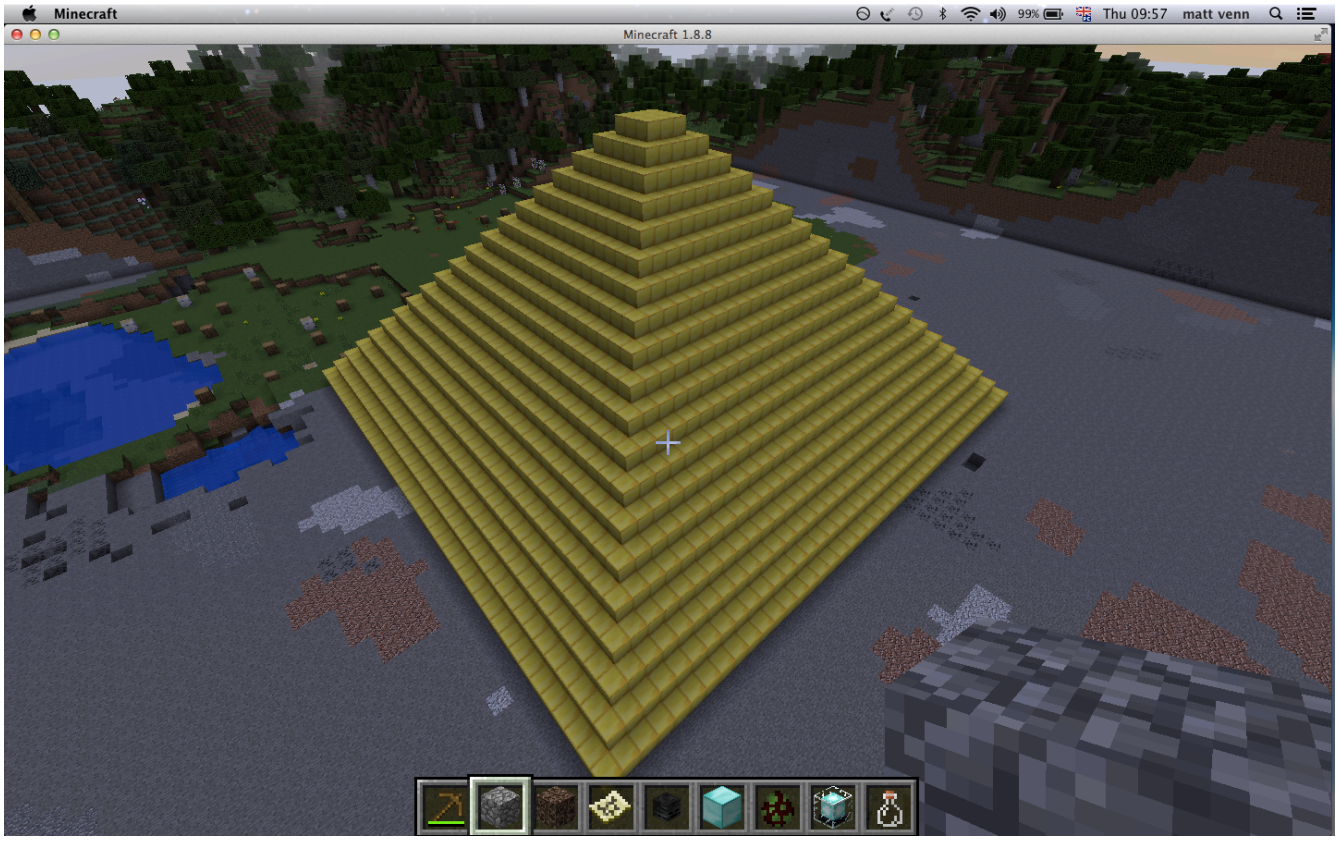
```
setBlocks(x1, y1, z1, x2, y2, z2, block_id)
```

- `x1` and `z1` are the same, starting at -5 and increasing by 1 each time,
- `x2` and `z2` are the same, starting at 5 and decreasing by 1 each time,
- `y1` and `y2` are the same, starting at 0 and increasing by 1 each time.

Try running this program - does it create the right numbers?

```
width = 5
height = 0

while width > 0:
    print(-width, height, width)
    height = height + 1
    width = width - 1
```

Can you add an extra line that uses `setBlocks` to create a pyramid?



Pixel Art

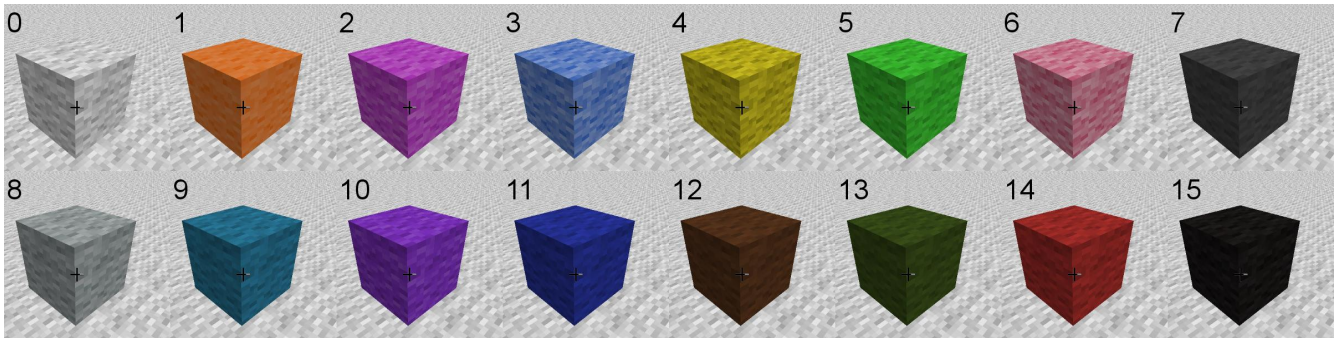
- This activity will help you understand data structures and lists in Python
- You will need: Minecraft with the Python API

You will draw a simple pattern on graph paper, and then write a program to reproduce this in Minecraft.

Draw your art

Get some graph paper or draw a grid on some plain paper. Use different colours in each block to build a simple image. Keep your image smaller than 8 x 8 blocks.

Then make a copy of your image, but convert the colours to numbers using this table:



Lists

Now convert your picture into a Python data structure - each row is going to become a list, and all the lists that represent rows will be put into another list that represents the whole picture.

This is the data structure used to create the picture above:

```
pixels = [
    [ 15, 13, 1, 14 ],
    [ 1, 3, 1, 14 ],
    [ 1, 13, 11, 14 ],
    [ 1, 13, 1, 15 ],
    ]
```

Each number represents one pixel. There are 4 pixels in a row and 4 rows in the whole picture.

You can test this works by iterating through the data structure. Iterating means stepping through it. In this case you'll be using a `for loop` because it runs the indented code *for* each item in the list.

```
for row in pixels:
    print("new row:")
    for pixel in row:
        print(pixel)
```

When you run the program you should see each value in the pixel data structure printed out.

Now change your program so that instead of printing out numbers in the shell, it creates the right coloured blocks in Minecraft.

To start with you'll need the usual stuff at the top of the program:

```
import mcpi.minecraft as minecraft

mc = minecraft.Minecraft.create()
```

Then add the `for loop` above. Run your program and check that you can see the right numbers being printed out in the right order.

Now you need to create the pixels by using the `setBlock` command to create `WOOL` blocks of different colours (like in the traffic light exercise).

You can do this by creating 2 variables, one for x position and one for y position.

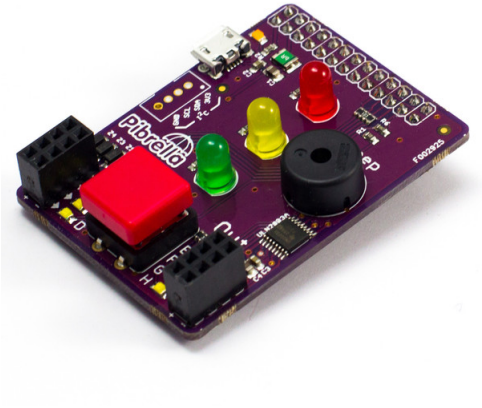
```
x = 0
y = 0
for row in pixels:
    y = y + 1
    x = 0
    for pixel in row:
        print(x, y, pixel)
        x = x + 1
```

Use this loop in your program, and add a `setBlock` command to create each pixel at the correct co-ordinates.

Treasure Hunt

- This activity will help you understand how to use a Pibrella with Minecraft
- The Vec3 position object
- Pythagoras theorum in 3D

You will use the PiBrella's lights to guide you to hidden treasure in Minecraft.



Lights

Type this program into a new file and save it as `treasure.py`:

```
import pibrella
pibrella.light.on()
```

Afer saving the program, you'll have to run it using the sudo command because accessing the Raspberry Pi's GPIOs needs super user privileges.

Start a terminal by double clicking the `LXTerminal` icon on the desktop. Then in the black terminal window type:

```
sudo python treasure.py
```

Which should turn all the PiBrella's lights on.

Calculating distance in Minecraft

In previous exercises, you've used `mc.player.getTilePos()` to find your position. This function returns a type of object called `Vec3`. In the past you've accessed the x, y and z co-ordinates like this:

```
if playpos.x == -247 and playpos.y == 10 and playpos.z ==60:
    # do something
```

This time you'll use another `Vec3` object to help calculate the distance between you and the treasure. Add this to your program:

```
from mcpi.vec3 import Vec3
# hide the treasure
destination = Vec3(100, 5, 20)
```

The `destination` variable is a `Vec3` object, and you can subtract different `Vec3` objects to find the difference between them.

Add this to your program:

```
import mcpi.minecraft as minecraft
mc = minecraft.Minecraft.create()

playpos = mc.player.getTilePos()
diff = playpos - destination
print(diff.x, diff.y, diff.z)
```

You should see the x, y and z distance between your current position and the treasure.

Pythagoras

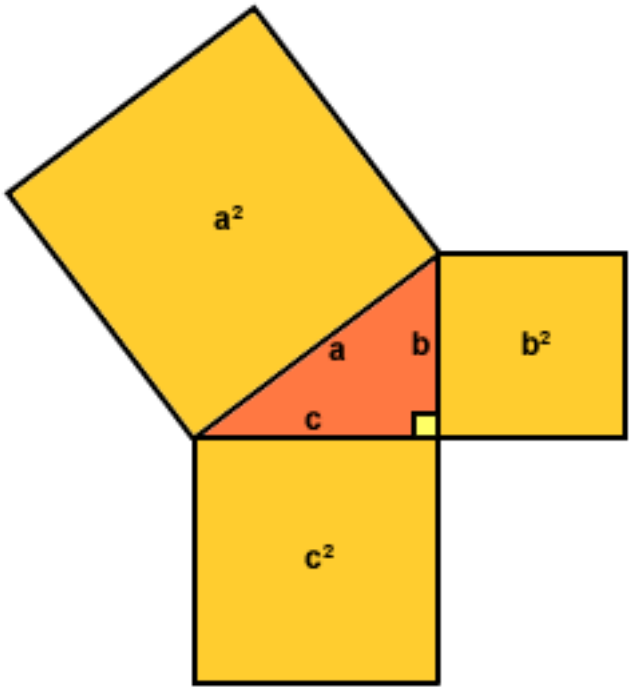


Figure 4: pythag

In the 2 dimensional world, you'll have come across Pythagoras' theorum. There is something very similar for the 3D world of Minecraft:

```
from math import sqrt
dist = sqrt(diff.x * diff.x + diff.y * diff.y + diff.z * diff.z)
```

Flashy lights

If you put your code in a loop, then as you move around in Minecraft, the distance will be constantly recalculated.

How can you make the lights flash faster the smaller `dist` is? Here's some code that turns a light on and off at a set frequency:

```
import pibrella
import time

while True:
    pibrella.light.red.on()             # turn on the red LED
    time.sleep(0.1)
    pibrella.light.red.off()            # turn off the red LED
    time.sleep(0.1)
```