Python and Minecraft

Contents

Big buildings

- KS3
- This activity will help you understand positions in Minecraft and loops in Python.
- You will need: Minecraft with the Python API

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

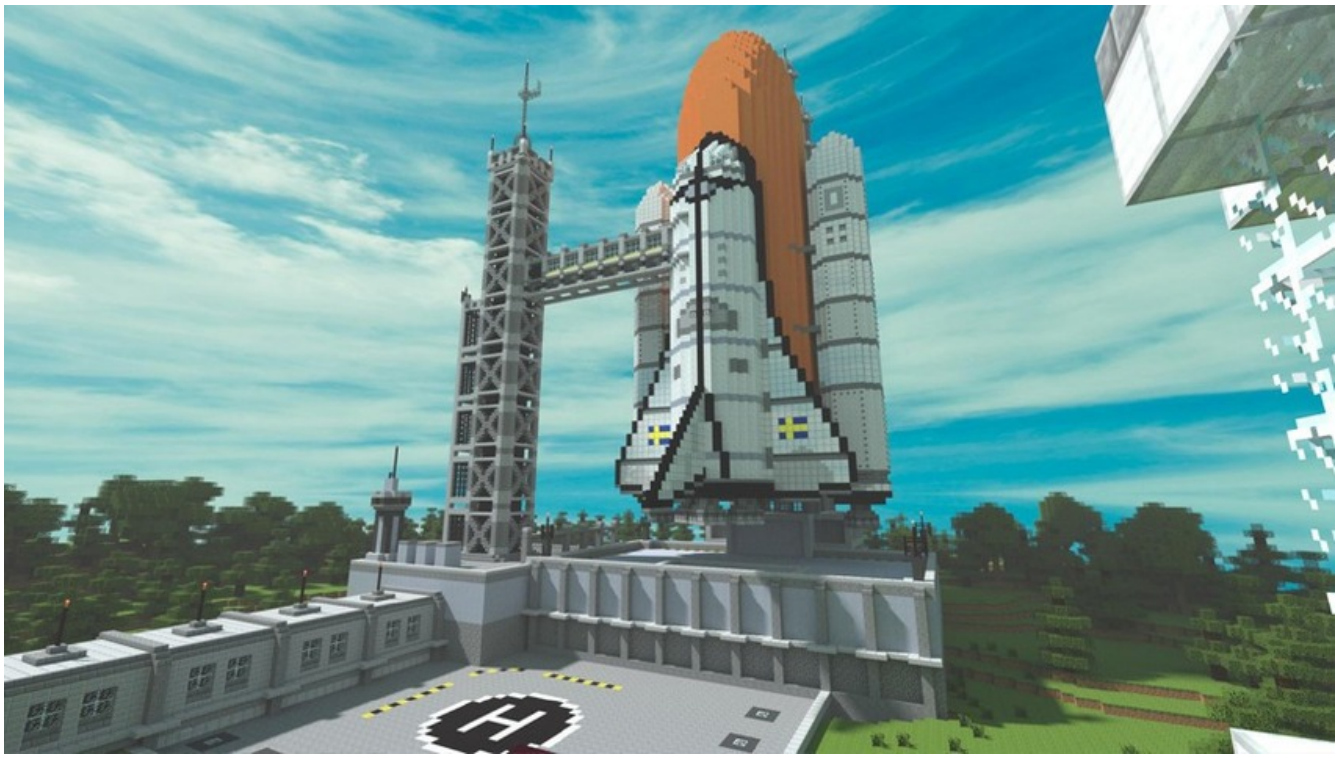


Image courtesy of crpeh

Single blocks

It's easy to create a single block in the Minecraft world. First of all, import the Minecraft libraries:

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

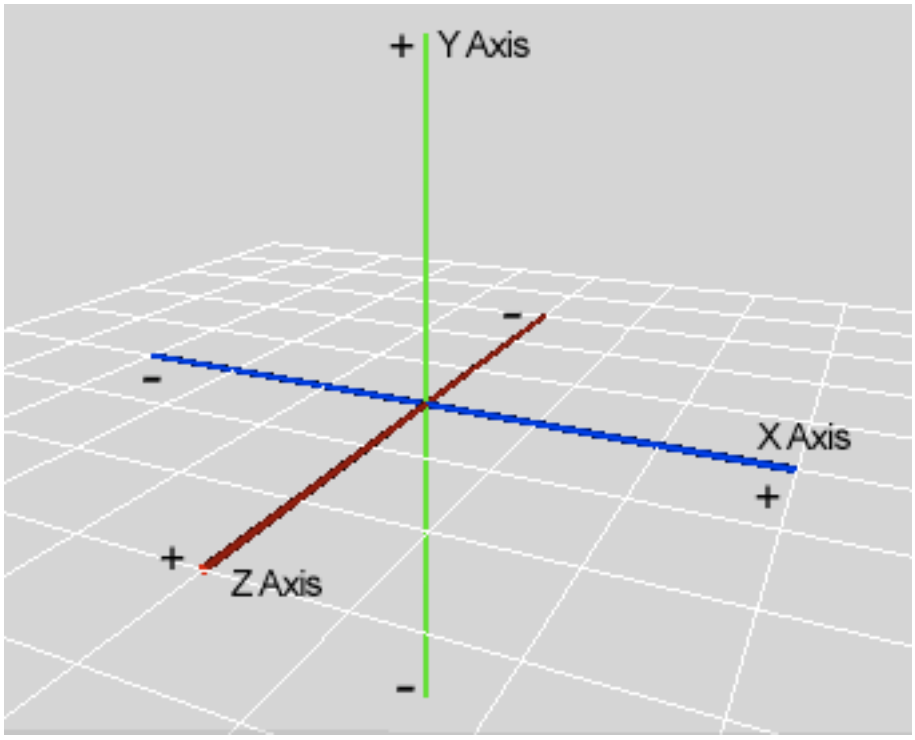
Then create a `handle` that will let you interface Python with Minecraft:

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

Then to place a block, use the `setBlock` code:

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

X, Y and Z are the co-ordinates where the block is placed:



And the `block_id` is either the number of the block you want to place, or its name. Both these commands make a gold block:

```
mc.setBlock(x, y, z, 41)
mc.setBlock(x, y, z, block.GOLD_BLOCK.id)
```

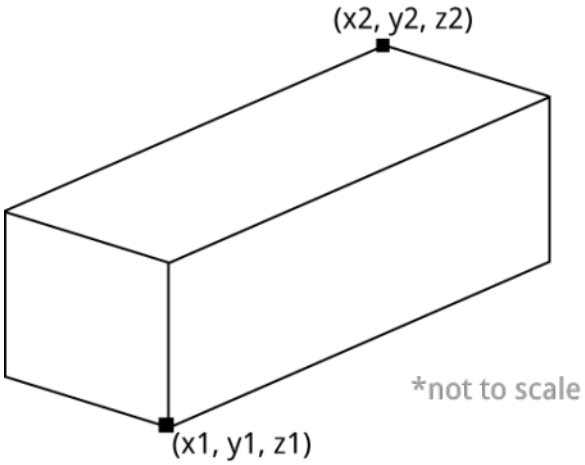
If you want to place a block where you currently are, you can look at the top left of the screen, where your X, Y and Z position is displayed.

Bigger blocks

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

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

The volume between x1, y1, z1 and x2, y2, z2 will be filled with blocks of type `block_id`.



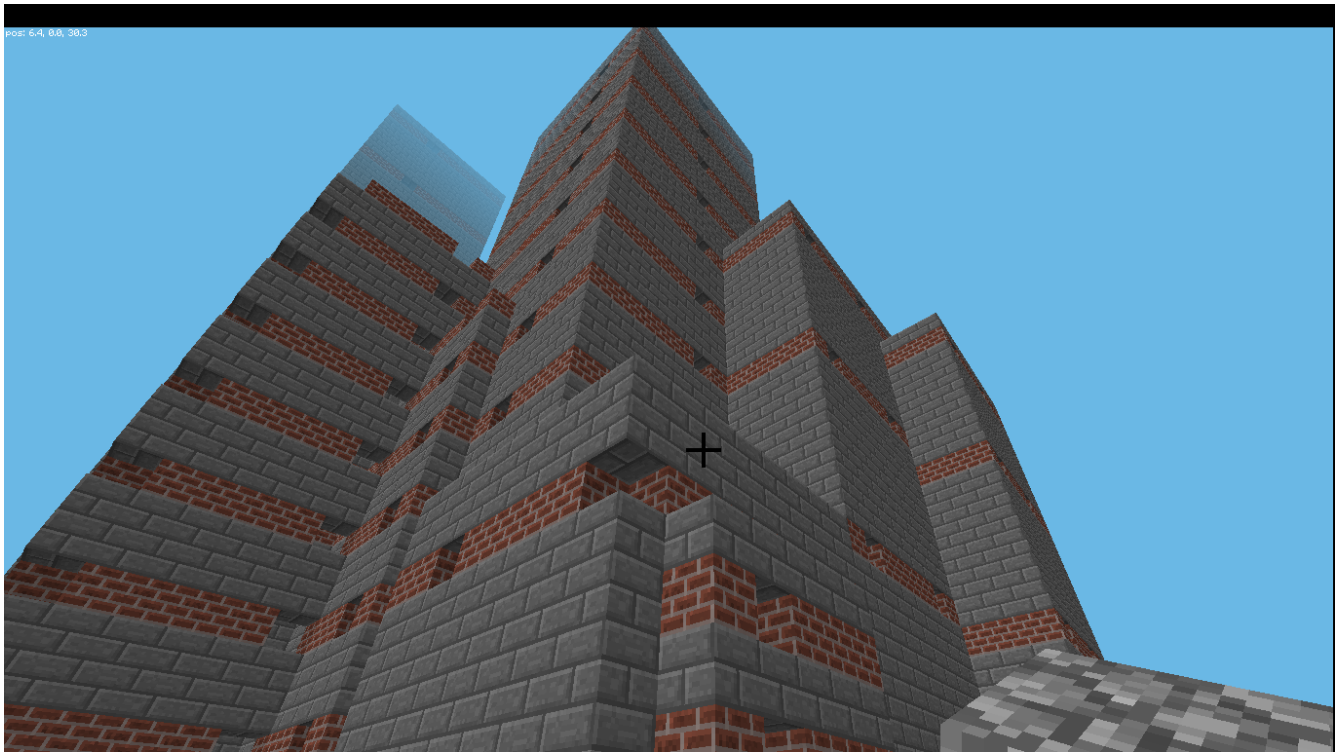
For more information about `setBlocks`, see this page by Craig Richardson

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

```
mc.setBlocks(-10,0,-10,10,10,10,block.AIR.id)
```

So when you clear the area with the command above, the volume is from x = -10 to x = 10, y = 0 to y = 10, and z = -10 to z = 10.

Tower blocks



Try using your `setBlocks` skills to build a few big tower blocks.

Pyramids

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

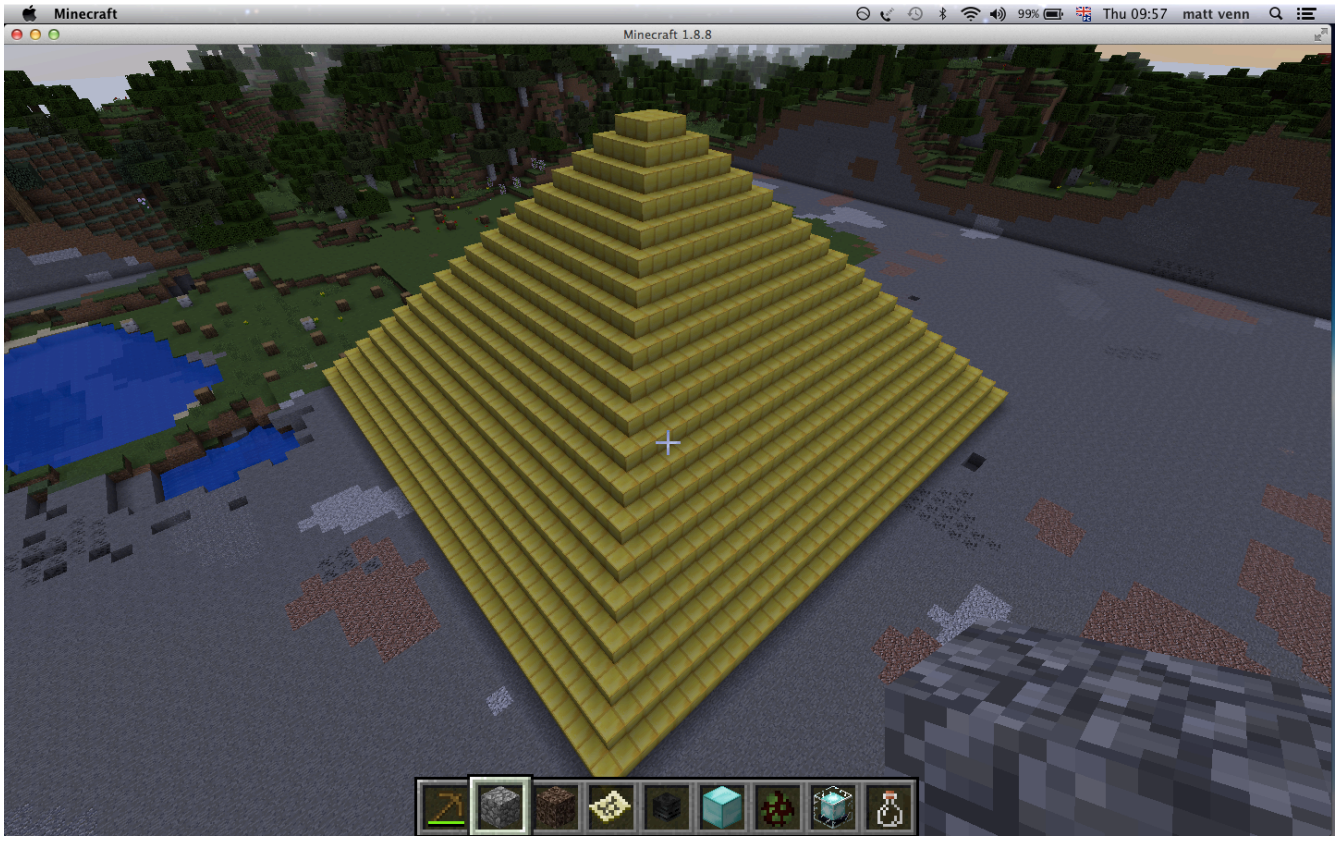
```
mc.setBlocks(-5,0,-5,5,1,5,block.GOLD_BLOCK.id)
mc.setBlocks(-4,1,-4,4,2,4,block.GOLD_BLOCK.id)
...
```

Can you see a pattern in the X, Y and Z numbers? Because there is a simple pattern, you can save time, or build bigger pyramids by using a loop:

```
width = 5
height = 0

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

Try using a loop to build a huge pyramid!



Traffic lights

- KS3
- 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 Steve there:

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

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

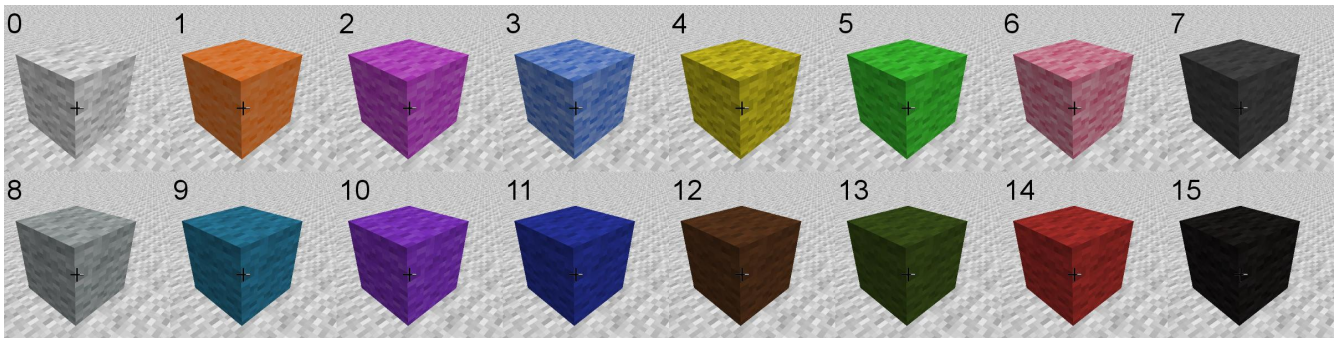
Either use a few `setBlock` commands or a single `setBlocks` command to create a traffic light.

Coloured wool blocks

Then create the 3 lights with a wool block set to black:

```
mc.setBlock(x, y, z, block.WOOL.id, 15)
```

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



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(x, y, z, block.WOOL.id, 14)
    # wait for 1 second
    time.sleep(1)

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

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

Trap!

- KS3
- This activity will help you understand positions in Minecraft and conditional statements in Python.
- You will need: Minecraft with the Python API

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 Steve's 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 Steve is 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

- KS3
- This activity will help you understand positions in Minecraft and conditional statements in 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 Steve's 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 Steve is 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)
```

Pixel Art

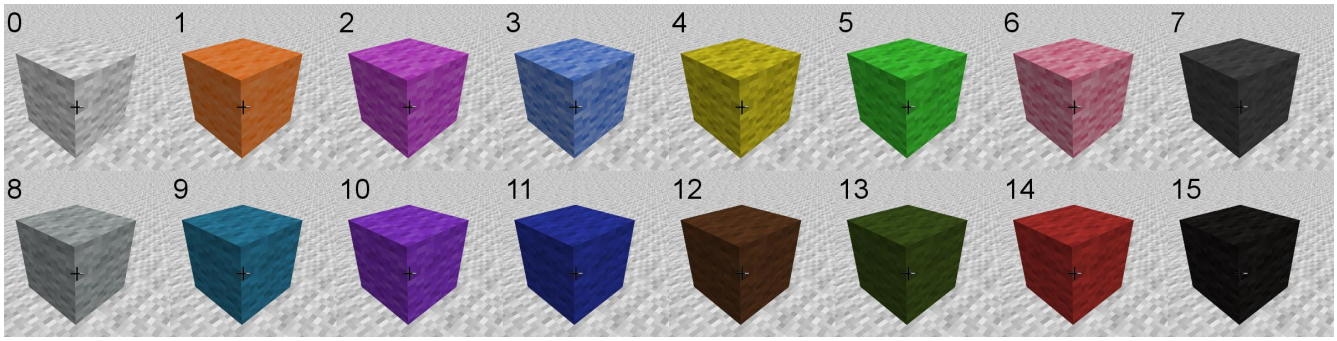
- KS3
- This activity will help you understand datastructures and lists in Python
- You will need: Minecraft with the Python API

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 ],
    ]
```

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
import mcpi.block as block

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

Then add the `for loop` above, modifying it so that the blocks get created in the right places.

Teleport

- KS3
- 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 Steve to a random location in Minecraft.

Button

Try this program to check the PiBrella is working:

```
import pibrella

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

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. If your program is called teleport.py, open a terminal, change to the directory where your program is and type:

```
sudo python teleport.py
```

Random numbers

To go to a random location you can make use of 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 Steve to a random new location.

Treasure Hunt

- KS3
- 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 lights to guide you to hidden treasure in Minecraft.

Lights

Try this program to check the PiBrella is working:

```
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. If your program is called treasure.py, open a terminal, change to the directory where your program is and type:

```
sudo python treasure.py
```

Calculating distance in Minecraft

In previous exercises, you've used `mc.player.getTilePos()` to find Steve's 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 Steve 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

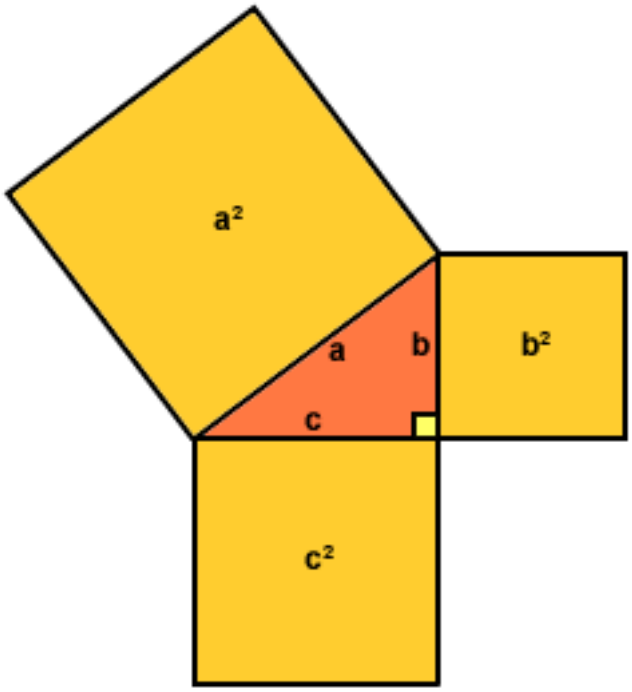


Figure 4: pythag

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)
```