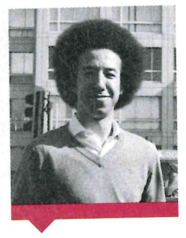
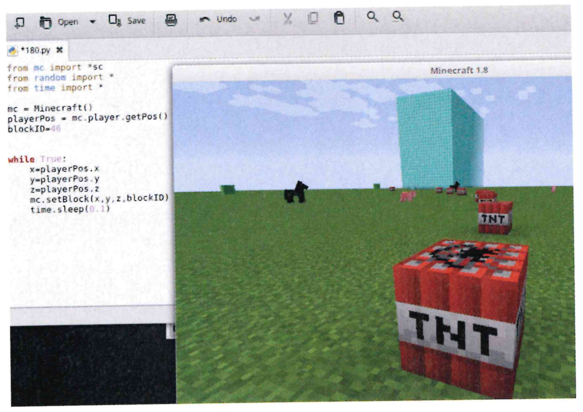
Control the player and environment in Minecraft

Continuing our series on Python coding in Minecraft, this issue we're looking at controlling the player character and the world around them



Calvin Robinson is head of computing and network manager at an all-through state school. Specialising in computer science, Calvin also works with schools all over London as a computing consultant in education, helping provide high-quality computing teaching and learning.



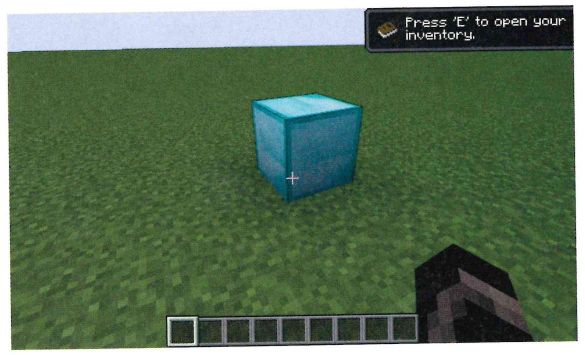
```
from mc import *sc
from random import *
from time import *

mc = Minecraft()
playerPos = mc.player.getPos()
blockID=46


while True:
    x=playerPos.x
    y=playerPos.y
    z=playerPos.z
    mc.setBlock(x,y,z,blockID)
    time.sleep(0.1)
```

What you'll need

- Minecraft **https://www.mojang.com/games**
- Python **https://www.python.org**
- McPiFoMo **http://rogerthat.co.uk/McPiFoMo.rar**
- Minecraft Turtle **http://bit.ly/MinecraftTurtle**
- Block IDs **http://bit.ly/minecraft_id_links**

Hooking directly into Minecraft with Python gives us many advantages, but rather than building by hand, we can use programming techniques such as loops to simplify much of the building process. In this issue's tutorial, we will manipulate the player character's position and the blocks around them. We use simple lines of code to replicate blocks in rows and columns, to build entire rooms out of thin air. We also teleport Steve or Alex around your world at the drop of a hat. Combining these two functions enables us to have a little fun with TNT to create our own bomber-man.

As a prerequisite, we assume you've installed McPiFoMo, from our last two issues. McPiFoMo includes MCPiPy by 'fleap' and 'bluepillRabbit' of MCPiPy.com; the Raspberry Jam Mod, developed by Alexander Pruss; and Minecraft Turtle by Martin O'Hanlon of www.stuffaboutcode.com.



01 Moving the player character

We can directly control our character's position using x,y,z coordinates. Create a new Python script and import mc.

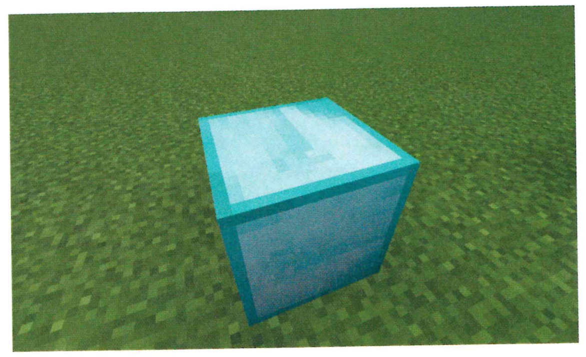
```
from mcpi import minecraft
mc = minecraft.Minecraft.create()
```

```
mc.player.setTilePos(20, 20, 20)
```

Of course, we can replace these values with variables, to make things easier later on:

```
x=20
y=20
z=20
mc.player.setTilePos(x, y, z)
```

Now whenever we want to change the player's position, we simply alter the x,y,z variables.



02 Placing blocks

As well as moving the player character, we need the ability to place and move blocks around our world. To do this, we use the setBlock command:

```
mc.setBlock(30,30,30,57)
```

Again, variables make life easier, so in practice we'd probably arrange it like this:

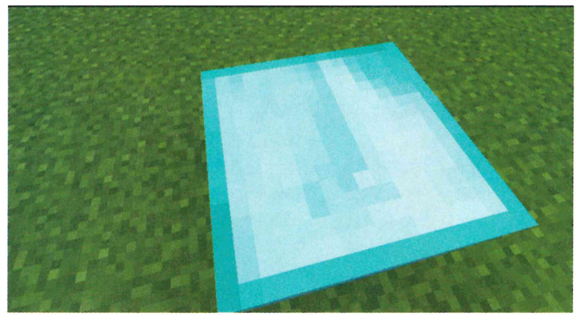
```
x=30
y=30
z=30
blockID=57
mc.setBlock(x, y, z,blockID)
```

The blockID for diamond is 57 – you can find more using the link in the What you'll need box (p72).

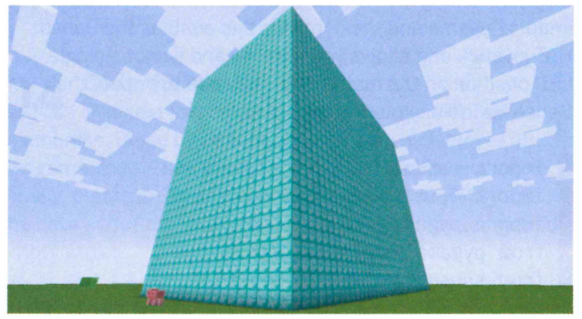
03 Combining the two

If we assign the player position to a variable, we can use it as coordinates for placing blocks, so that we can create something around our player.

```
playerPos = mc.player.getPos()
x=playerPos.x
y=playerPos.y
z=playerPos.z
mc.setBlock(x,y,z,blockID)
```



This will place a block exactly where our player is. You can then offset the coordinates accordingly: x+=1. Remember: x is East/West, z is North/South and y is Up/Down.



04 User input to define variables

With the addition of a typical input command, we can ask the user what type of block they'd like to create.

```
blockID = input("Which blockID would you like to use? ")
```

Taking that a step further, let's ask the user where they'd like the block(s) placed:

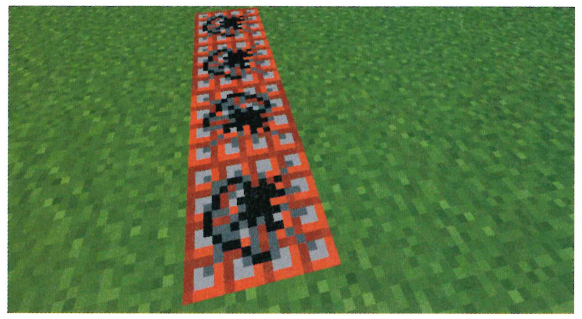
```
relativeX = input("How many blocks East of your position would you like this block placed? ")
mc.setBlock(relativeX,y,z,blockID)
```

You can of course do the same for the y and z coordinates.

05 Build in rows and columns

Placing blocks one at a time can slow down the process. There may be instances where we want to place entire rows and columns of blocks at once. After setting your playerPos and blockID (as in **Step 3** and **Step 4**), we need to add variables for rows and columns.

```
eastWest = 20
northSouth = 30
upDown =  40
mc.setBlocks(x,y,z, x+eastWest, y+northSouth, z+upDown, blockID)
mc.setBlocks(x+1,y+1,z+1, x+eastWest -1, y+northSouth -1, z+upDown -1, 0)
```



06 Explosion man

Now that we're familiar with playerPos and setBlock, we can combine the two to turn your player into a walking bomber-man. By moving our earlier code into a while loop, and with the addition of a simple Boolean, we can make our player drop bombs with every step:

```
while True:
        playerPos = mc.player.getPos()
        x=playerPos.x
        y=playerPos.y
        z=playerPos.z
        mc.setBlock(x,y,z,46)
        time.sleep(0.1)
```

Now it's time to have some explosive fun!

Take your time

Whether you're placing blocks or moving your player character around the world, you may want to space out your commands. Using Python's `time.sleep` command, we can create an artificial gap in our code.

For instance, `time.sleep(10)` would pause the program for ten seconds before initiating the next line of code. This could be useful if you wanted a certain block to appear every x number of seconds, or if you fancy moving a player around at specific intervals.

In order to use `time.sleep`, you will need to `import time` at the top of your code.