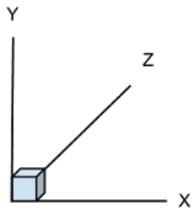
MineCraft is cool because you can easily build things in a virtual world. What if you could create a program to make something instead of doing it yourself, block by block? The Raspberry Pi edition of Minecraft allows you to do just that using the Python programming language! (I'm assuming you already have MineCraft setup for the Pi and running, and are in the correct directory*.) Launch the Python IDLE editor from the desktop, and then use the interactive shell to import the modules that let you talk to the Minecraft server:

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

The Pi's MineCraft world is represented by a cube that is 256 blocks on each side. This is an important number in computing because it can be represented by 8 binary (0 or 1) "bits", which is also known as a "byte". In binary, 11111111 = 255 (you get to count the zero, so it is 256 numbers).



Each location within the Minecraft world is identified using an X, Y, Z coordinate system.
The designer wanted to limit the memory usage so that a position could be saved in three bytes of data.
You can determine your player's location within the world with the following command:

```
playerPos = mc.player.getPos()
```

That is a good thing to know, but how can we (the programmer) see it?

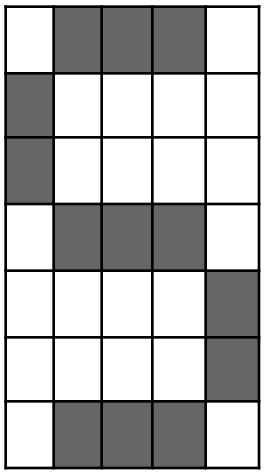
```
print(playerPos)
mc.postToChat(playerPos)
```

Now that we know where we are, let's place a diamond block nearby.

```
mc.setBlock(playerPos.x,
            playerPos.y + 2,
            playerPos.z - 3,
            block.DIAMOND_BLOCK)
```

Do you see it? If you didn't get an error, it is there somewhere. Place another in a different location. If you place your cursor at the end of the previous command and hit "Enter", it will place it on the current line for you to modify (or use Alt + p).

How about we create the first initial of your name using blocks? The player is known as "Steve", so I'll show you how to do an "S". We could use the `setBlock()` command for each location, but that would take a lot of typing. We should let the computer do the work for us! A very basic font can be created from a grid of 5 x 7 like this:



If we convert each of the shaded cells to a “1” and the white cells as “0”, we can represent the letter as:

```
S = ['01110',
     '10000',
     '10000',
     '01110',
     '00001',
     '00001',
     '01110']
```

Since the 1 and 0 are bits, and we are mapping where they go, this is literally a “bit map”!

Now for some code to loop through the bitmap and place some blocks for us (anything after the “#” is just for information and doesn’t need to be typed):

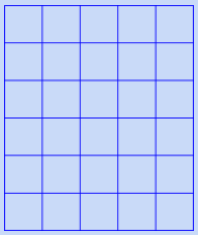
```
x = playerPos.x + 10                # start a bit away from us
y = playerPos.y + 7                 # and up in the air
z = playerPos.z - 10                # and over to the side

for line in S:                      # a string like '01110'
      for character in line:        # each bit in the string
            if character == '1':
                  mc.setBlock(x, y, z, block.REDSTONE_ORE)
            z = z + 1               # move one space over
      y = y - 1                     # move one space down
      z = z - len(line)             # reset horizontal location
```

Did it work? If not, double check your typing and try again.

Explore:

Use the grid and create the first letter of your first name. Convert it to a bitmap and show it in minecraft like we did above.



```
_ = ['_ _ _ _ _',
     '_ _ _ _ _',
     '_ _ _ _ _',
     '_ _ _ _ _',
     '_ _ _ _ _',
     '_ _ _ _ _',
     '_ _ _ _ _',
     '_ _ _ _ _']
```

Minecraft Part 2: Creating Functions and Buildings

That seems like a lot of typing to get one letter displayed, but now we have the pattern down, we can reuse it over and over again by turning it into a reusable piece of code called a **function**. We should also save it to a file as well! In IDLE, select [File] -> [New Window]. Then place the following code in the new window and save it as something snappy, like "i_control_minecraft_now.py"!

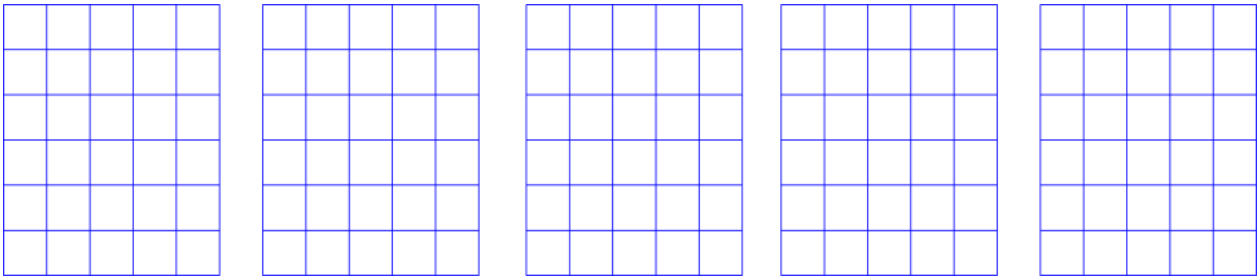
```
def showLetter(letter, x, y, z):
    '''Places blocks for a single letter.'''
    for line in letter:             # a string like '01110'
        for character in line:      # each letter in string
            if character == '1':
                blockType = block.REDSTONE_ORE
            else:
                blockType = block.AIR
            mc.setBlock(x, y, z, blockType)
            z = z + 1               # move one space over
        y = y - 1             # move one space down
        z = z - len(line)           # reset horizontal location
```

To run the code in the file, hit the "F5" key. You can then continue with the following commands in the interactive shell, or add them to the file to be run (when you hit F5 again):

```
letters = [S, T, E, V, E]
for letter in letters:
    showLetter(letter, x, y, z)          # call the function!
    z = z + 6                            # move over and add a space
```

Explore:

- Finish the letters of your name (or Steve's) and show them in world.
- Can you change the function to show the letters overhead like clouds? (hint: look at the x-y-z diagram above).
- How could you change the function to accept `blockType` as an argument like we do position?



You have now seen how to do loops, functions, and if/else branching. Can we use that knowledge to construct a building in world? Add the following to your file, or start a new one.

```
width = 10
depth = 10
height = 10
xStart = playerPos.x - 10
yStart = playerPos.y
zStart = playerPos.z
x, y, z = (0, 0, 0)

# floor
blockType = block.STONE
for x in range(depth + 1):
    for z in range(width + 1):
        mc.setBlock(xStart + x, yStart + y, zStart + z, blockType)

# walls
blockType = block.BRICK_BLOCK
for y in range(1, height):
    for x in range(depth): # left side
        mc.setBlock(xStart + x, yStart + y, zStart, blockType)
    for x in range(depth): # right side
        mc.setBlock(xStart + x, yStart + y, zStart + width, blockType)
    for z in range(width): # front
        mc.setBlock(xStart, yStart + y, zStart + z, blockType)
    for z in range(width): # back
        mc.setBlock(xStart + depth, yStart + y, zStart + z, blockType)

# roof
blockType = block.STONE
for x in range(depth + 1):
   for z in range(width + 1):
       mc.setBlock(xStart + x, yStart + height, zStart + z, blockType)
```

Explore:

- Can you add windows or doors (using Python, of course)?
- Could you turn the building code into a function so you could make multiple buildings?
- What else could you create or do in Minecraft using programming?

That is about as far as I'm going to take you for now. Time to strike out on your own. I hope you had fun and want to keep learning!

* To get MineCraft setup for your Pi, or to learn how to do an analog clock with blocks or play the "Hotter / colder" game: http://www.stuffaboutcode.com/2013/04/minecraft-pi-edition-api-tutorial.html

You also don't need a Raspberry Pi. If you know how to setup a MineCraft server, you can use Python with the regular MineCraft version:
http://mcpipy.wordpress.com/2013/02/13/running-python-programs-without-a-raspberry-pi/

Minecraft Block Types:

AIR	FURNACE_ACTIVE	MUSHROOM_RED
BED	FURNACE_INACTIVE	NETHER_REACTOR_CORE
BEDROCK	GLASS	OBSIDIAN
BEDROCK_INVISIBLE	GLASS_PANE	REDSTONE_ORE
BOOKSHELF	GLOWING_OBSIDIAN	SAND
BRICK_BLOCK	GLOWSTONE_BLOCK	SANDSTONE
CACTUS	GOLD_BLOCK	SAPLING
CHEST	GOLD_ORE	SNOW
CLAY	GRASS	SNOW_BLOCK
COAL_ORE	GRASS_TALL	STAIRS_COBBLESTONE
COBBLESTONE	GRAVEL	STAIRS_WOOD
COBWEB	ICE	STONE
CRAFTING_TABLE	IRON_BLOCK	STONE_BRICK
DIAMOND_BLOCK	IRON_ORE	STONE_SLAB
DIAMOND_ORE	LADDER	STONE_SLAB_DOUBLE
DIRT	LAPIS_LAZULI_BLOCK	SUGAR_CANE
DOOR_IRON	LAPIS_LAZULI_ORE	TNT
DOOR_WOOD	LAVA	TORCH
FARMLAND	LAVA_FLOWING	WATER
FENCE	LAVA_STATIONARY	WATER_FLOWING
FENCE_GATE	LEAVES	WATER_STATIONARY
FIRE	MELON	WOOD
FLOWER_CYAN	MOSS_STONE	WOOD_PLANKS
FLOWER_YELLOW	MUSHROOM_BROWN	WOOL

Here are the bitmaps to finish “Steve” off:

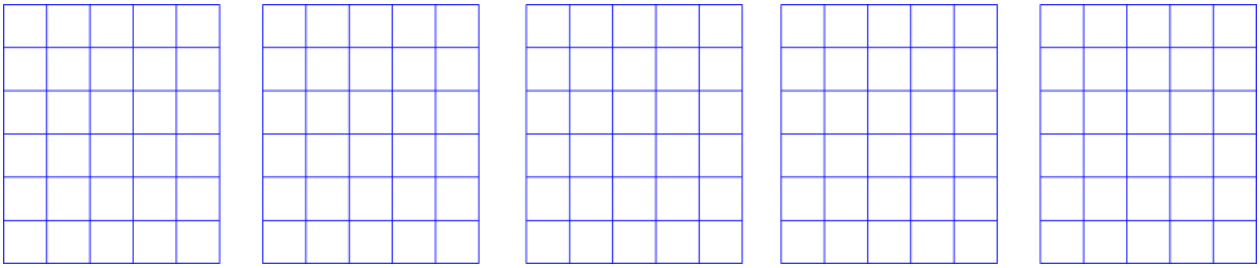
```
S = ['01110',
     '10000',
     '10000',
     '01110',
     '00001',
     '00001',
     '01110']
```

```
T = ['11111',
     '00100',
     '00100',
     '00100',
     '00100',
     '00100',
     '00100']
```

```
E = ['11111',
     '10000',
     '10000',
     '11110',
     '10000',
     '10000',
     '11111']
```

```
V = ['10001',
     '10001',
     '10001',
     '10001',
     '10001',
     '01010',
     '00100']
```

More 5 x 7 bitmaping grids:



#i_control_minecraft_now.py template

```
# Download MineCraft for Raspberry Pi from http://pi.minecraft.net/
# Script assumes mcpi/api/python/mcpi folder copied to /usr/lib/python2.7/dist-packages

import mcpi.minecraft as minecraft
import mcpi.block as block
from time import sleep

mc = minecraft.Minecraft.create()

def showLetter(letter, x, y, z):
    '''Places blocks for a single letter.'''
    for line in letter:             # a string like '01110'
        for character in line:      # each letter in string
                if character == '1':
                    blockType = block.REDSTONE_ORE
                else:
                    blockType = block.AIR
                mc.setBlock(x, y, z, blockType)
                z = z + 1           # move one space over
        y = y - 1                   # move one line down
        z = z - len(line)           # reset horizontal location

playerPos = mc.player.getPos()

x = playerPos.x + 10
y = playerPos.y + 2
z = playerPos.z - 1

S = ['01110',
     '10000',
     '10000',
     '01110',
     '00001',
     '00001',
     '01110']

letters = [S, S, S]
for letter in letters:
    showLetter(letter, x, y, z)            # call the function!
    x = x + 6                              # move over and add a space
```