Minecraft Workshop
Part 3

Using Functions to Simplify Your Code

```
1  from mcpi.minecraft import Minecraft
2  import mcpi.block as block
3  mc = Minecraft.create()
4
5  def main():
6      while True:
7          pos = mc.player.getPos()
8          x = pos.x
9          y = pos.y
10         z = pos.z
11         stacks(x, y, z)
12
13 def stacks(i, j, k):
14     stacker(i-3, j, k-3)
15     stacker(i-3, j, k+3)
16     stacker(i+3, j, k-3)
17     stacker(i+3, j, k+3)
18
19 def stacker(a, b, c):
20     mc.setBlocks(a, b, c, a, b+5, c, block.DIAMOND_BLOCK.id)
21
22 main()
```

There is a more detailed explanation in the 'How Functions Work' sheet but essentially our code runs as follows:

1. Import the modules we need, and create a connection to Minecraft.
2. Define a function called **main** all of the indented code on the following lines is part of this definition and only gets run when we call the function using **main()**.
3. Define a function called **stacks**.
4. Define a function called **stacker**.
5. Call the function named **main**. Which means that the code inside **main** will now run.
6. Start the loop inside **main**, get the player's position, store it in each of the three variables.
7. Call the function **stacks** giving it the current position of the player.

8. The program now runs the code inside **stacks** using the coordinates we called it with. This code in turn calls the function **stacker** four times, passing it slightly different coordinates on each call.
9. Each time the code in **stacker** is run, the program then **returns** to where it was called from, ie inside **stacks**. It then moves to the next line of code and runs that.
10. When the program gets to the end of the code in **stacks** it will return to where **stacks** was called from, ie the last line of **main**.
11. Since the code inside **main** is part of a loop, the loop then runs again (the next **iteration** of the loop) and everything begins again at step 6 above.

Now see if you can work through each of the following challenges in order to recreate as accurate a representation of the Giant's Causeway as possible using Minecraft (we will have to make square stacks rather than hexagonal ones unfortunately).

- **Bronze Challenge:** Change one of the earlier programs you created to make it use functions.
- **Silver Challenge:** Change the code above to place 9 random height stone stacks near to you
- **Gold Challenge:** Change the code so that it only places the stacks if they are on top of stone.
- **Platinum Challenge:** Change the code so that when you are stood in the middle of the peninsula on the Giant's Causeway Map it will place random height stacks above any existing stone blocks
- **Diamond Challenge:** Change the code so that it detects when the stack will be near to water and ensures that the maximum possible height is reduced (this should make the stacks look more realistic).
- **Ultimate Challenge:** Make the stacks look more realistic by using blocks such as `MOSS_STONE` and anything else you can think of.