**Etude 4: Coloured Cubes**

**Problem**

Given a 2x2x2 cube made out of a combination of yellow and blue 1x1x1 cubes (which are entirely blue or yellow), find every possible unique 2x2x2 cube, taking into consideration rotational symmetry.

**Process**

When we first looked at this problem we thought we could just visualise/make a cube and rotate the cube and look for unique arrangements but we quickly realised that this was a tedious task and was very prone to error hence there is no way we could ensure or justify that we have found every unique cube combination. We decided to programmatically solve this.

The first issue we encountered was how to turn a 3D cube into data so that we could compare and look for arrangements. We settled on a 2D array where each subarray holds a “level” of a cube, either the top 4 cubes or the bottom 4 cubes. Due to their only being two possible colours we used “0” to represent yellow and “1” to represent blue. Now that we had the data structure we could create two functions to rotate the cube. We did this by exchanging values of particular indexes so that it mimics the two types of rotation required – right rotation, and up rotation.

We knew that 82 is 128 (eight 1x1x1 cubes that can be one of two colours) so we needed a way to make 128 unique combinations of 0’s and 1’s to represent every arrangement of the cube (without taking into consideration rotation and symmetry). Because we have only two states, 0 and 1, we can use a binary representation to represent every possible arrangement of cubes (e.g. [ [0,0,0,0], [0,0,0,1] ] represents 7 yellow small cubes and 1 small blue cube on the bottom layer) and we store all these in an array.

Once we generated all 128 cubes we then test each of them against a dictionary where we hold every unique cube. Because we have to check every rotation against every other rotation, we do the following in order to ensure that we find and compare every cube:

* For each side (4 times), rotate **right** (along the x-axis)
* For every right rotation we then rotate **up** for each side (4 times) (along the y-axis)
* And for every up rotation we rotate **right** again for each side (4 times) (along the x-axis)
* Then we see if that arrangement of the cube (array) is in the unique cube dictionary and if it is then we break out of the loops and move onto the next cube (without adding it to the dictionary), this is so the only way cubes get added to the dictionary is if it has cycled through every possible rotation and still hasn’t been found in the dictionary (meaning it hasn’t been found and counted yet).

Note: these rotations are stacked in nested loops so for every right rotation do 4 up rotations etc.

Because we rotate the cube in every possible way and compare each one to the dictionary, we can be sure that we have found all the unique arrangements of the cube considering rotation and symmetry. We found that there are **23** unique cubes (the length of the dictionary) which can be shown by running the code included.