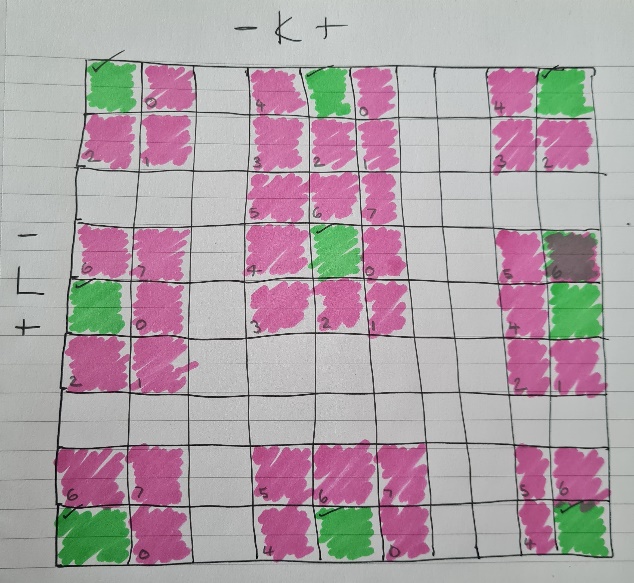
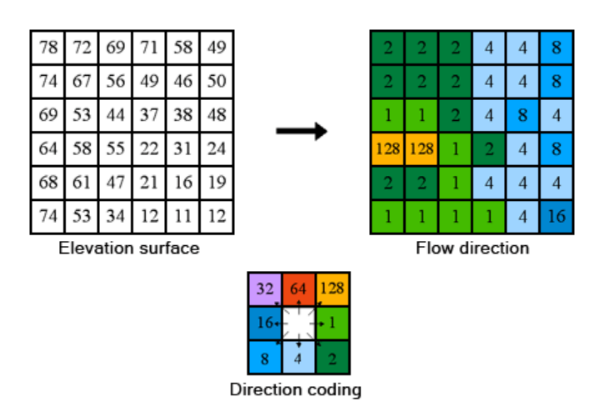
**Development Document**

To construct this program the desired functionality was broken down into pseudocode at the beginning. This mostly consisted of breaking down the possible values of l and k to categorise each potential pixel as either a corner, a side, or a centre. Difficulties were encountered when writing the if conditions for the 3 x 3 centre pixels, so the decision was made to assign these to the else condition. To help visualise the conditions for each pixel, the matrix was drawn out and each neighbouring pixel was assigned a pixel number from 0 – 7 (See **Figure 1)**. Even numbered pixels were orthogonal to the original pixel (green) and Odd numbered pixels were diagonal.

A whichpix function was created to identify the pixel with the largest decrease in elevation from the original pixel. Then this information was passed to the flow direction function which used a formula obtained from the literature (Callaghan, 1984; Jenson & Domingue, 1998) along with information obtained from ArcGIS Pro Tool References **(See Figure 2).**

*Max\_drop = elevationdeduct / math.sqrt(2) \* 100 (if a diagonal)*

*Max\_drop = elevationdeduct / 1 \* 100 (if orthogonal)*

Then, an update function updates the original pixel with the directional code value returned from the flow direction function. From code encountered in the previous agent-based-model task, contour-filled graphical outputs were coded. One was placed early in the program, after the elevation data had been read to generate a digital elevation model output. The second was placed at the end of the program after the bottom right pixel directional code had been assigned. This generated a D8 Directional Flow output. As the dataset is categoric, not continuous like the original elevation data, displaying the data using this output was problematic. Similarly, difficulties were encountered with the level parameters for this output. The levels parameter allows you to choose the minimum and maximum values along with the increments they should be separated by. However, most of the values were between 1-10. The possibility of using a streamflow plot in matplotlib was considered however, the developer was short on time in this circumstance so decided on a workaround for this issue. The following code was written:

list\_to\_plot = []

for inner\_list in elev\_copy:

list\_to\_plot.append([math.log(x, 2) for x in inner\_list])

with a list comprehension to quickly change the directional flow values for display purposes from [1, 2, 4, 8, 16, 32, 64, 128] to [0, 1, 2, 3, 4, 5, 6, 7] allowing them to be displayed as a contour-filled graphical output with a different colour assigned to each directional code value (See **Figure 3**)

