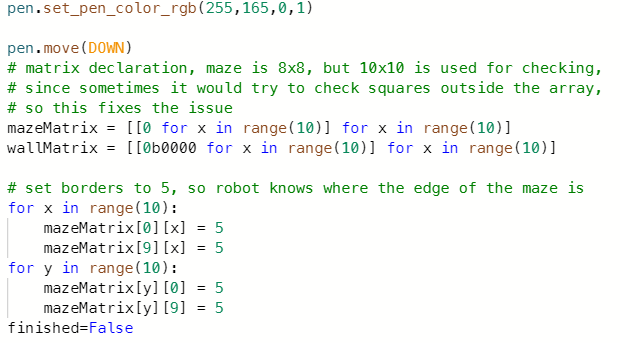
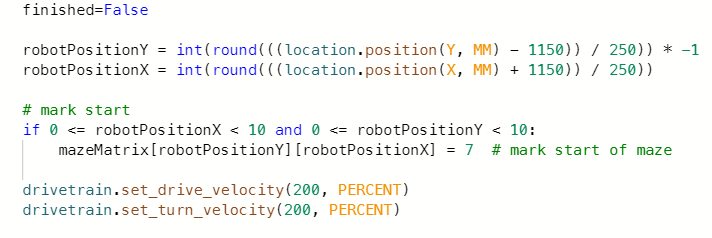
# Vex robot pathfinding documentation

My program works by first setting a pen down, to draw the robot’s path.

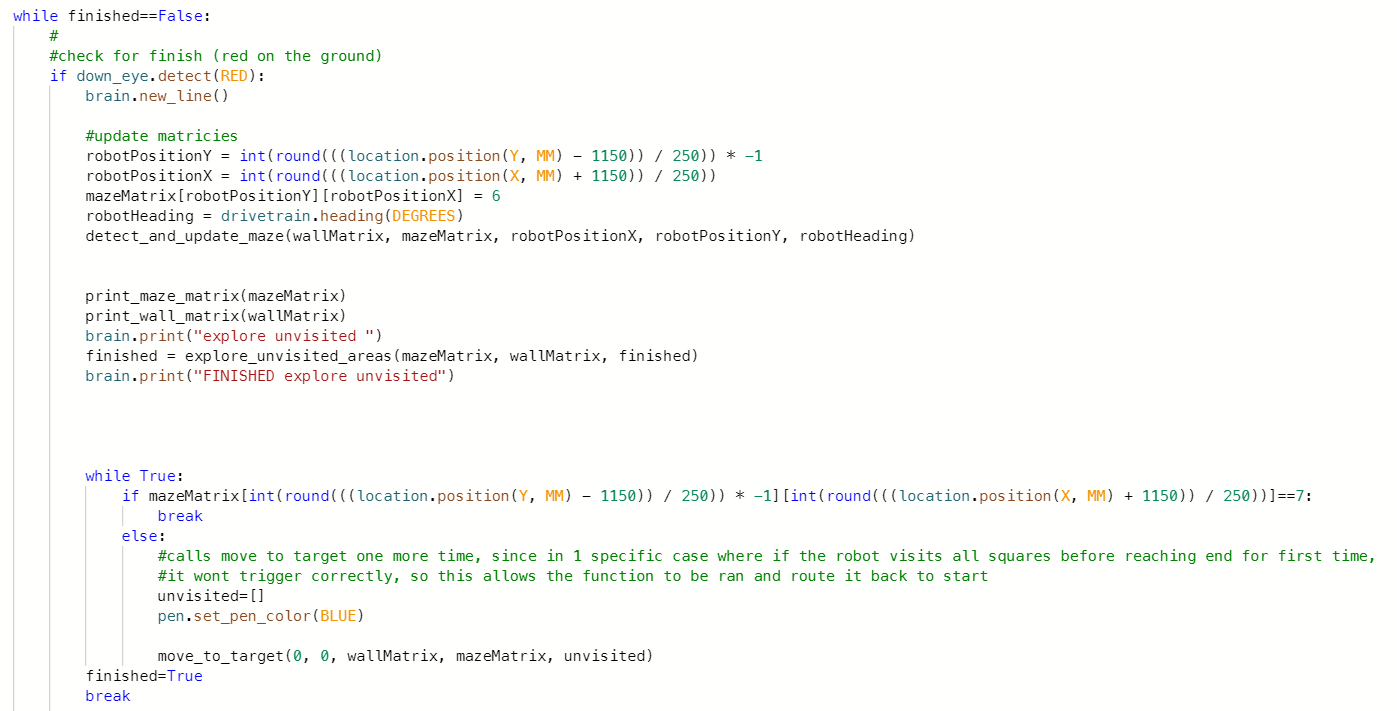
Next it continues by declaring 2 matrices

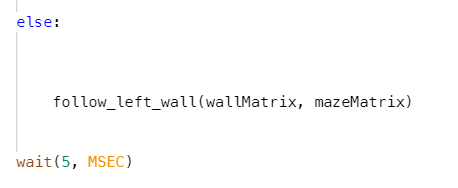


Then it continues with some basic variable declaration/setup, as well as getting the robots position inside the matrix using a basic formula, then goes into a while loop

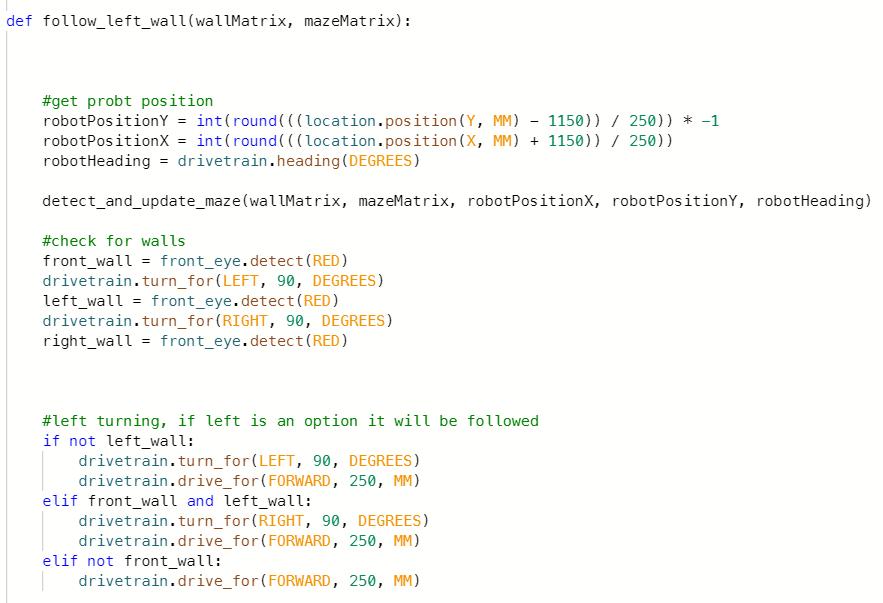


Inside this while loop there is a simple if/else, which checks if the robot is either at the finish of the maze or not

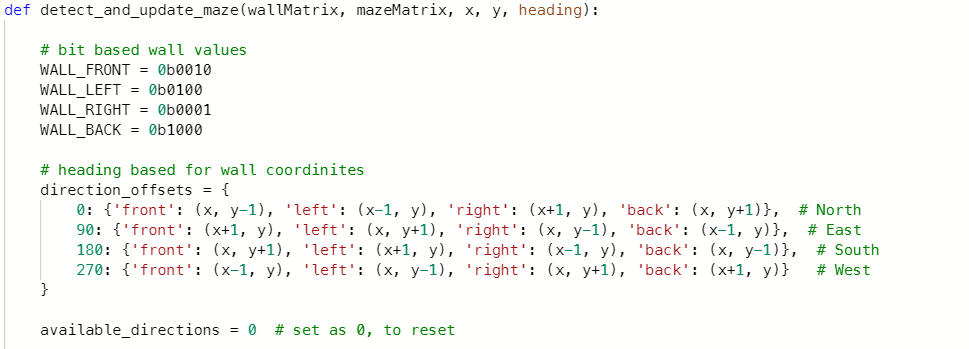




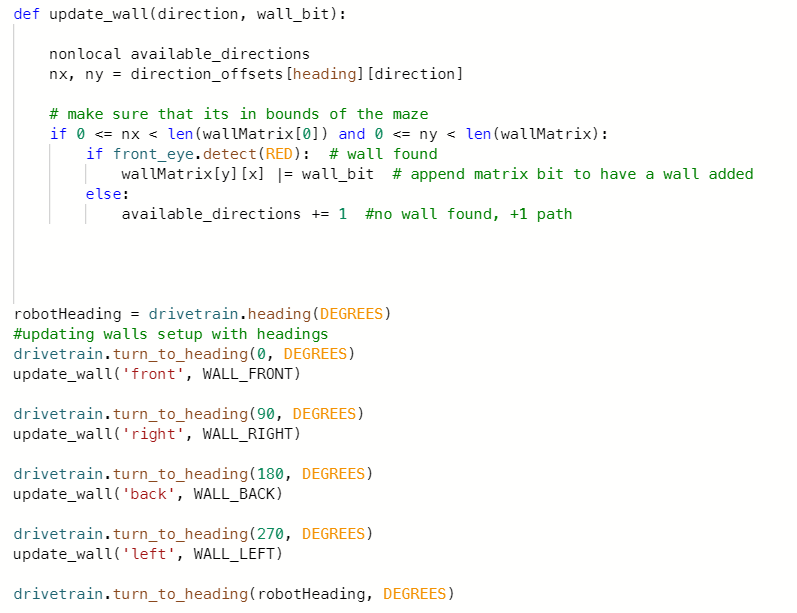
Since the robot is not on a red tile, it will activate the else statement, which then will run the follow left wall function



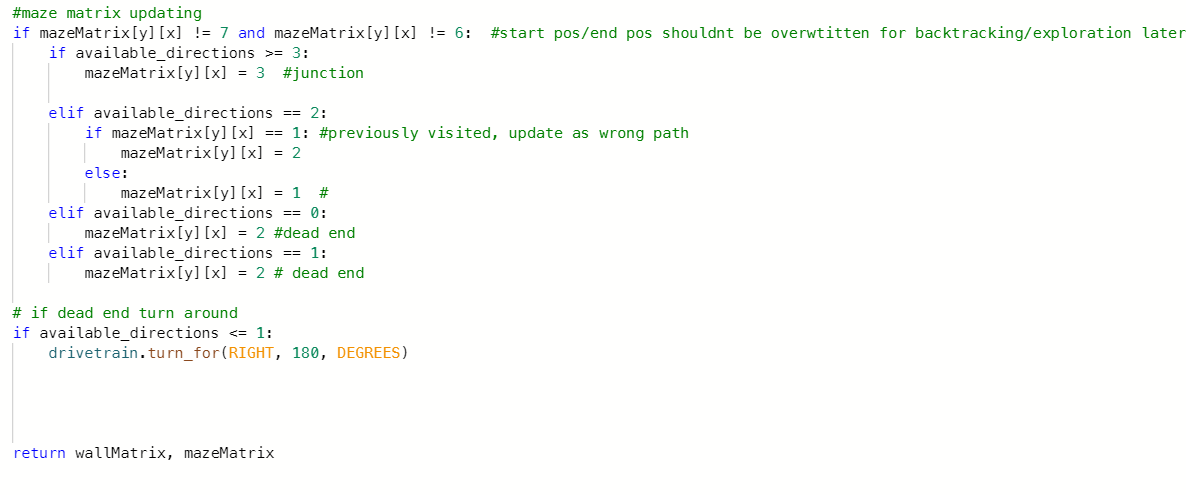
Inside this function the robot gets its position and heading then calls the detect and update maze function, which updates the wall matrix and maze matrix



Inside this function bit-based wall values are used, and the four ending 1/0s can be used to update the wall matrix, so 0b1111 would be walls on 4 sides, and 0000 would be no walls, and any combination of 1/0s works

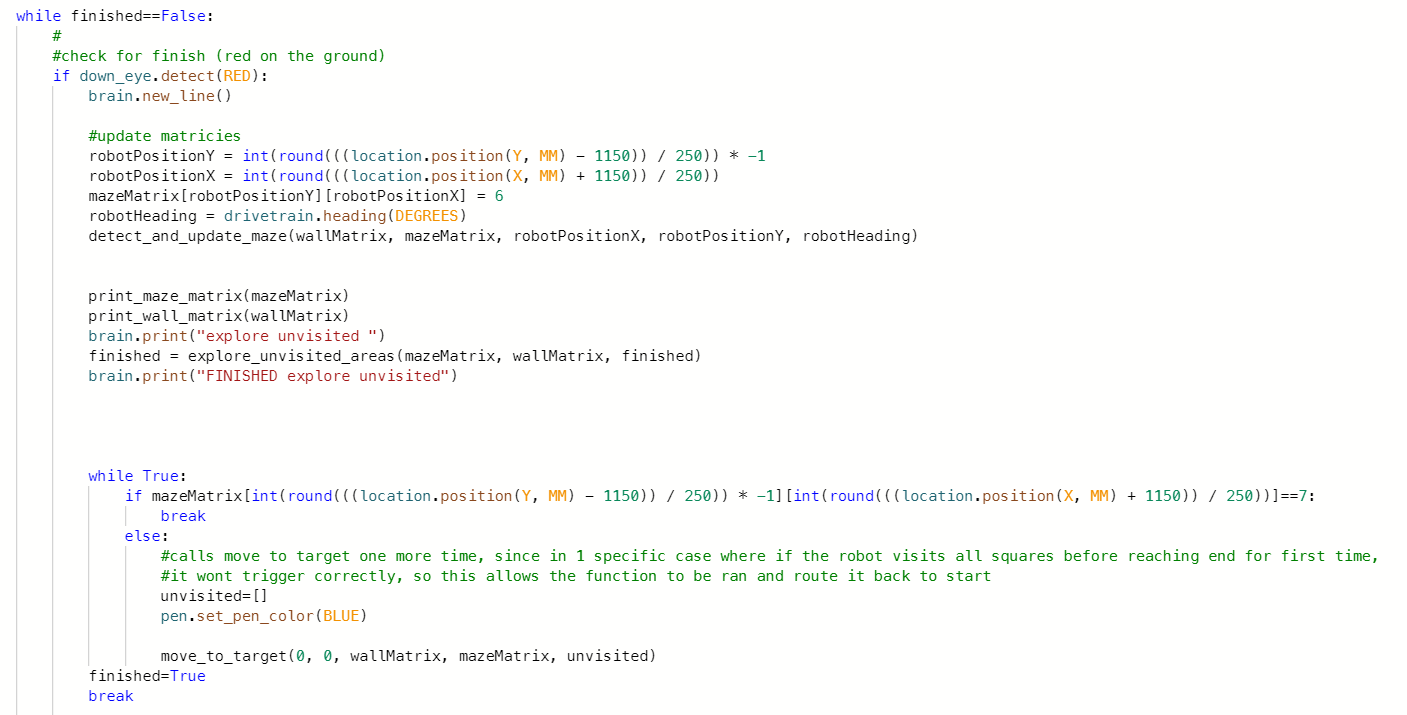


Next it continues by getting its heading, to restore its previous direction, then turning the robot in specific directions, and checking for how many directions are available as well as if the robot detects a wall, meaning that the bits will be appended



Then I go to updating the maze matrix, which will check the available directions to work out if it is either a junction, standard corridor, or a dead end, as well as checking if the path is already visited, then returns the wall and maze matrices to the main, causing the while loop to be checked again, this continues till the robot reaches the end/red tile

(while loop inside main)

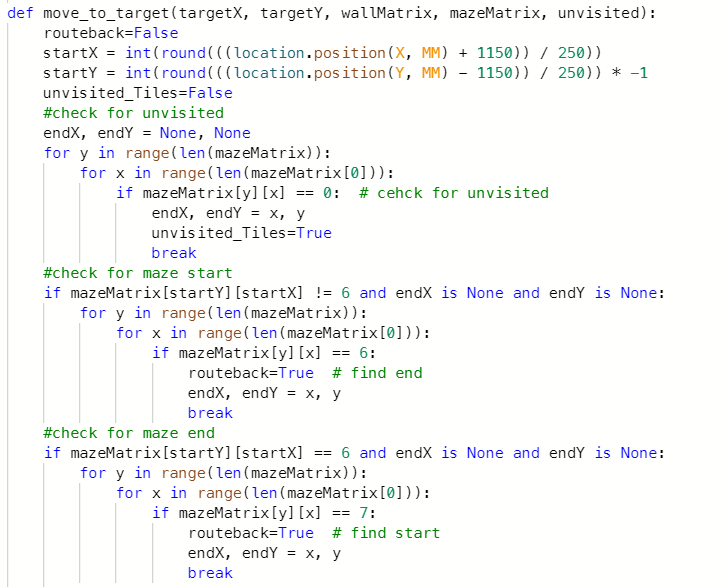


I continue by updating the wall and maze matrix on the finish tiles, using the same update functions from before, then go the explore unvisited areas function



In the function I start off by starting a while loop that checks if the maze is finished mapping, then create an array which contains unvisited positions, if all tiles are already visited, then the while loop will break

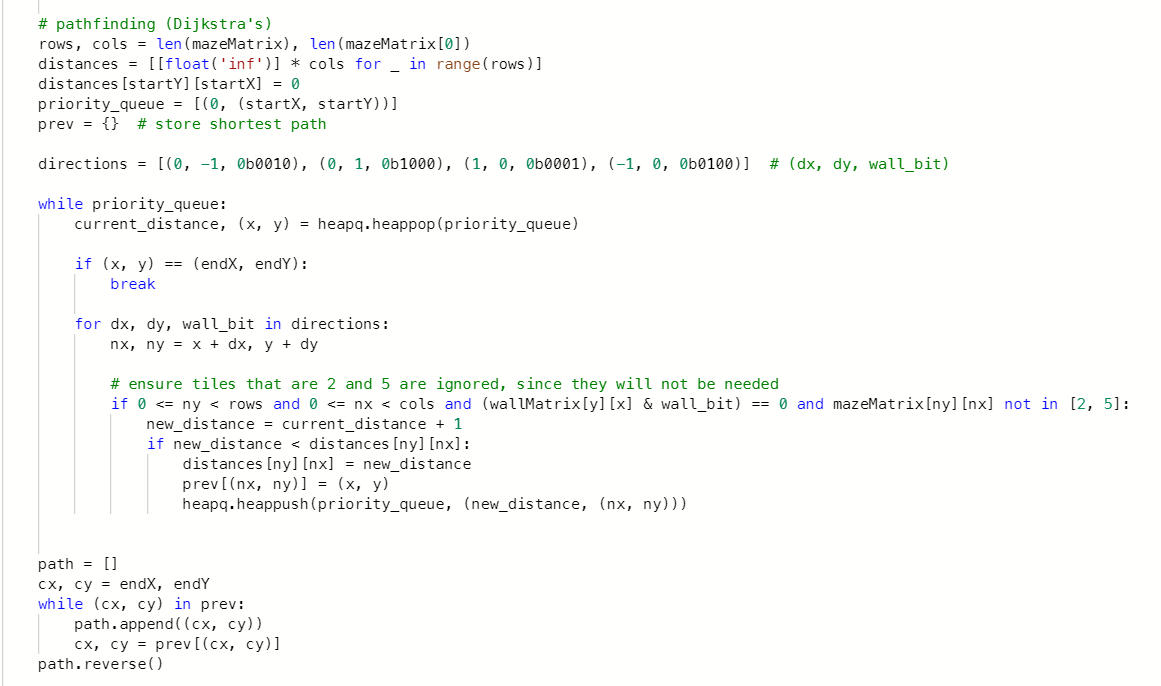
If this is not the case then the move to target function will be ran



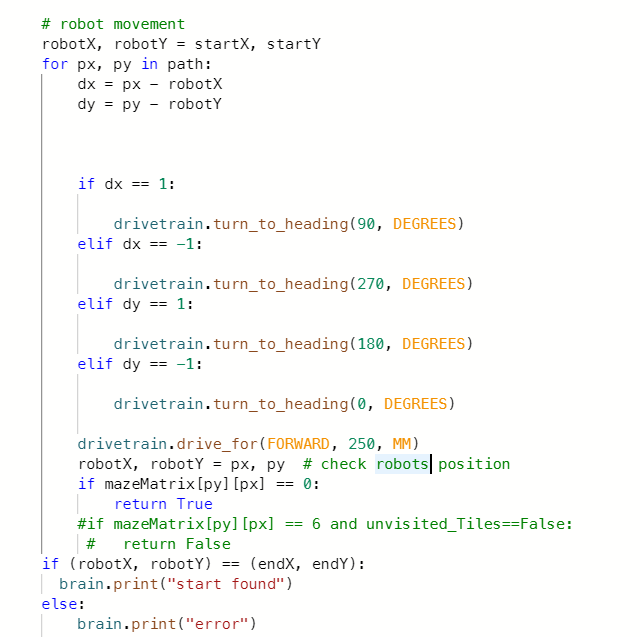
Next it continues by checking which unvisited tile it wants to visit, and has 2 if statements for navigating to the finish/start for once the unvisited tiles are completed.

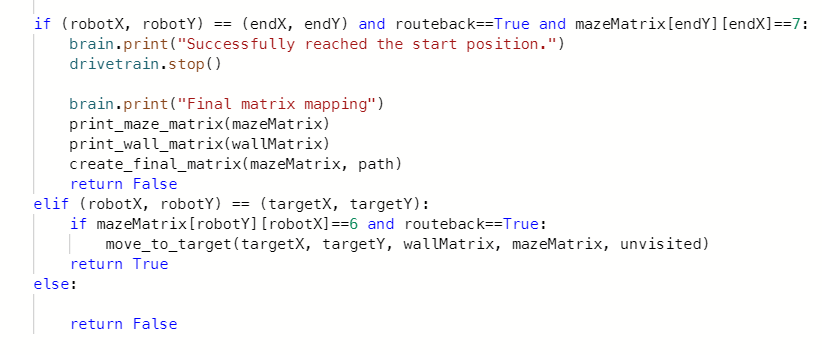
Then it continues using Dijkstra’s algorithm to route itself using the maze/wall matrix to find each unvisited tile





Generates shortest possible path using maze/wall matrices

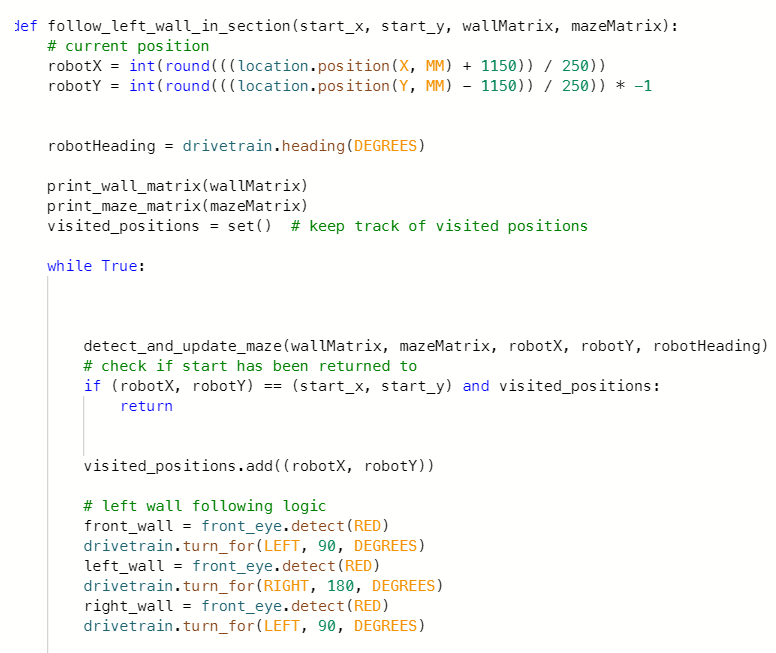


Then this section follows the path that has been generated

Which then follows into this section which will return true and allow the explore unvisited areas function to continue, after returning true to found\_tile, causing the if statement to run



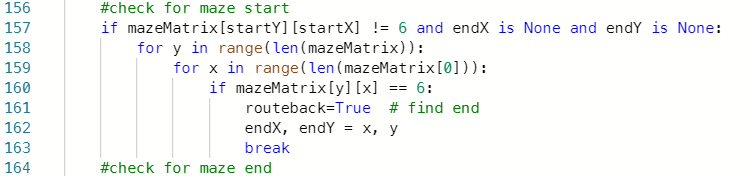
Inside this if statement it checks the robots heading, and appends the robot position based off this, to allow the follow left wall in section function to work correctly





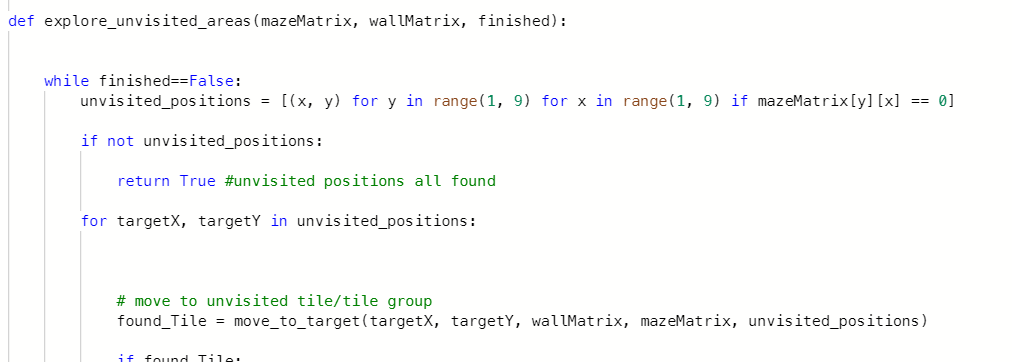
This here starts off by detecting and updating like before, then follows the left wall like before, the difference here is its set to terminate at a specific point, when the robot is back at where the left wall following started

It continues with this process until the maze is fully mapped, then the move to target function is called one more time, and then will find that all spaces have been mapped, then will move to the if statements here, and will route itself to the finish position, which is marked as 6 in the maze matrix

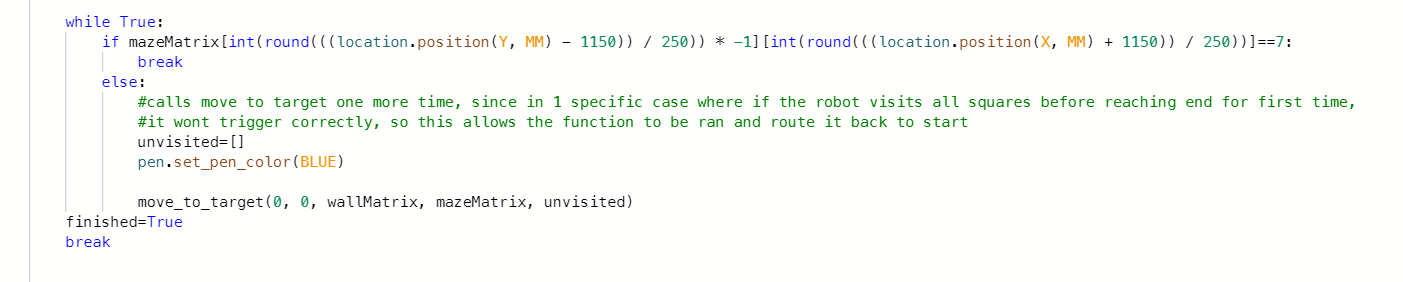


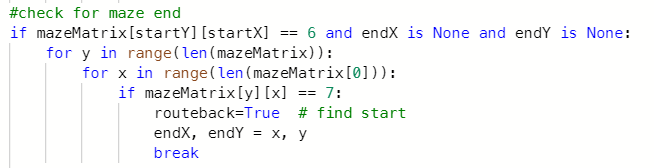
This causes the robot to go back the finish, using the Dijkstra’s algorithm from before

Then it returns true to explore unvisited areas, which will cause it to exit the explore unvisited areas function and return to main



Then inside main it will change the pen colour, to show a clear routing of the mazes fastest path

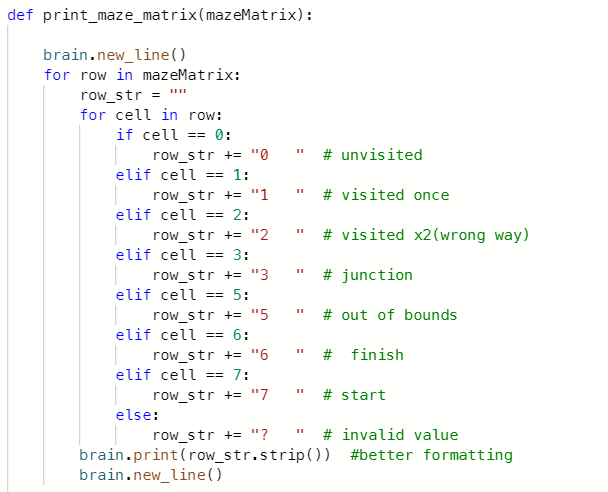
then when runs the move to target function one more time, which then will end up loading this if statement, since all are true



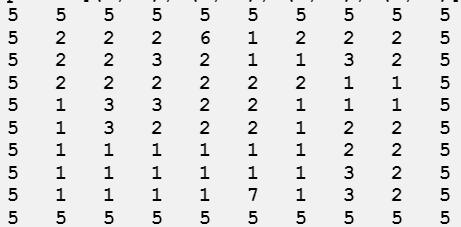
And this will route the robot back to the start, returning true and making the while function inside main return true, since the robot is back at the start and ending the program

I also have some prints inside my program which print 3 arrays in specific ways,

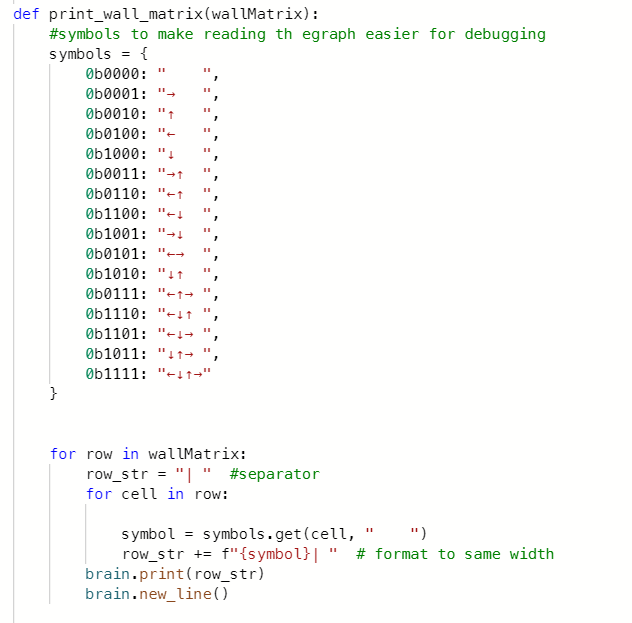
The mazematrix is printed using this



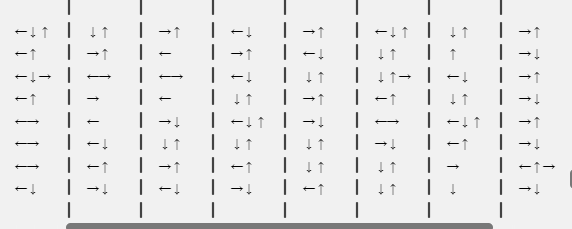
It prints it using good spacing for clear readability



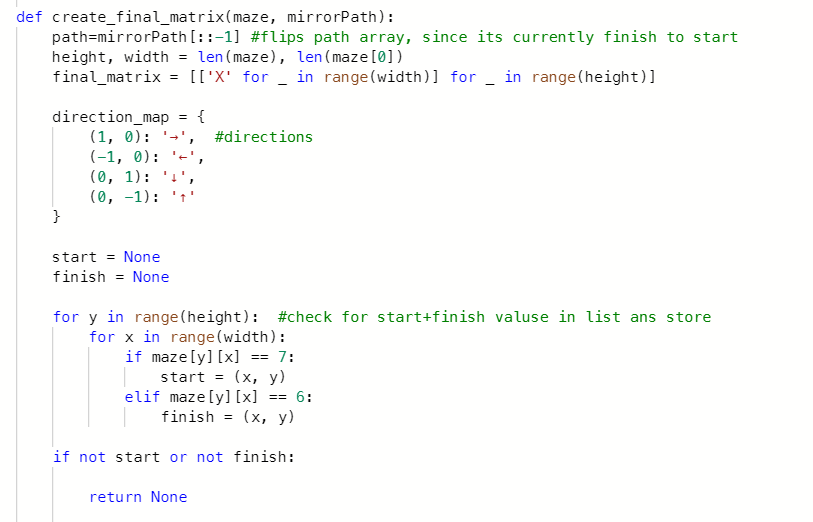
Then I print the wall matrix using this

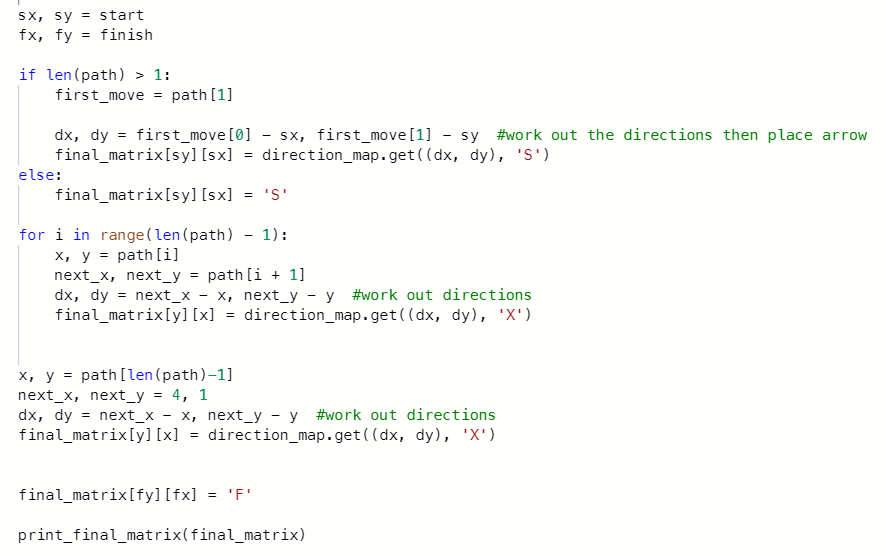


This appends the numbers of the wall matrix and makes the readable arrows, which is much better to read for bug fixing

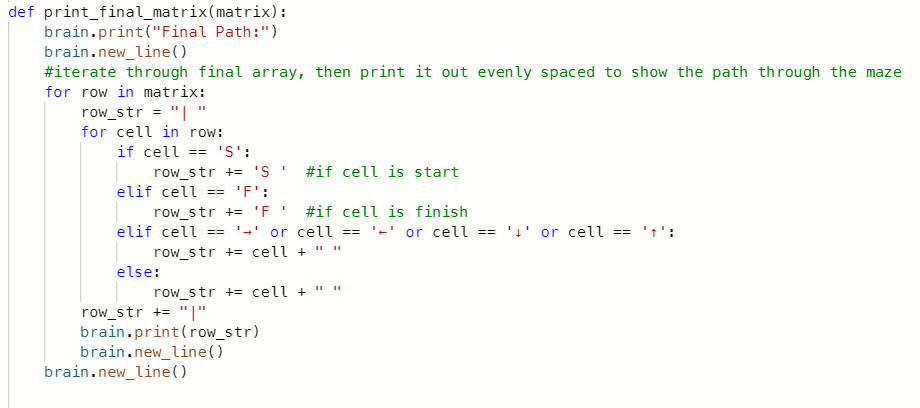


Then I continue and generate and print the final matrix, which shows the way out of the maze start to finish





What this does is it checks the direction that the robot will have moved based off the path, and the angle of the robot, and uses this to place the arrows that show the direction the robot follows



Then I print it out and then make sure to print arrows to show the directions, F to show the finish and “X” to show incorrect tiles

