Calum Lynch Assignment 1 Code Explanation

I start by importing all modules I need

In the part under comment #Task 1 I ask for two inputs in the form of two comma separated values (e.g. 10,15), split them with the delimiter ",", assign them to a and b to change the type to integers and then assign them to the variable start as a list to be used as coords. This is repeated for the end coords.

In the section under the comment #Task 2 the user is asked to input the file name (including file type) as a string. This fie is then opened in read mode and assigned to the variable roads. The script then assigns each line of the file as an element to the list M where each string is then stripped to remove white space and end of line characters and then split with the default delimiters to give a list of lists in the correct format for the np.array function. The list M is then converted to a numpy array with the data type int32 and assigned to the variable city_map.

Below #Task 3 I begin by defining the function check_bounds which checks the input coord_pos is less than the output of file_len (either the length or width of city_map depending how you look at it) or less than zero. If the if statement succeeds then a sys.exit is used to close the program. Next file_len is defined. It goes to the first line of the input file and then counts the lines and assigns that to l which is then returned. Finally check_bounds is called for all 4 starting coord inputs.

Within Task 4 the function is_valid is defined. It uses an if statement to check if the corresponding values in city_map for fc_pos and n_pos exist such that fc_pos > n_pos and that n_pos is not in fpath. If this check passed the function returns 1 else, it returns 0.

Task 5 was started by using the function find_path to set up for the recursive function procedure. The function starts by defining c_pos, c_path, path, and output as global functions s that they can be called and changed by other functions. Then it sets these variables to their starting values and calls procedure. procedure is a recursive function that updates the path and current path (c_path), checks for exit conditions, calls move then calls itself. Move is a function that calls is_valid for the function north (with the argument c_pos) in the last argument space. If is valid returns 1 then move returns north(c_pos) else it repeats this for east, south and west. If none of these is_valid calls return 1 then move calls backtrack and assigns the output to c_pos then returns. The function north(position) copies the coordinates that are input, changes them such that its one move north then calls perodic_boundary and assigns the output to fpos and then returns fpos. The functions east, south, and west do the same thing but in their respective directions. The function perodic_boundary checks if either of the coords input are larger than the result of file_len -1 or less than 0 and if they are then it will subtract or add the result of file_len respectively to make the program have periodic boundaries. The function backtrack removes the last element from c_path, if c_path is empty then it sets output to false and returns which then exits procedure and thus lets the script finish. Otherwise it sets the local var pos to the last element of c_path, removes that from c_path and returns pos. find_path is called which then calls all the other functions within Task 5.

For Task 6 the value of output is used to determine if the no suitable paths statement or the path has xx steps should be printed.

Task 7 only prints something if the output was True and in that case it uses for loops and iterates over the length and width of the matrix. If the position lines up with one of the sets of coords in c_path then the index of that coord from the list c_path is printed, otherwise "__ " is printed.

The file roads is then closed.