# Step 1: Analyze the Problem

We are tasked with designing a script to output information about a DEM raster file from the Open Data Areas Alberta Project. We are to output various information about the DEM such as: how many points there are in the file, what is the lower-left coordinate, what is the upper right coordinate, the approximate dimension of the file in kilometers, the minimum and maximum elevation value. To do this we will take the file location of the DEM file from the user and have python read each line in the file. This way I can extract and split each value and use it as I need it. Once I have all the information I need stored as different variables I can output it to the user in a neat and tidy format within the terminal. The basic idea to get the min/max values is to read each line and split it so that each value is separate. The program will then do this for each line and as it goes through if the value it see is less than/greater than the previous/original value then it will be replaced.

# Step 2: Design a Solution

Pseudocode:

# File Setup, reading, and splitting  
Get the file location from the user  
Read the file  
Split the file so that each set of X, Y, Z coordinates is on one line

# count the number of points in the file  
set up number of points counter  
for each line of coordinates in the file:  
 add to the number of points counter

# split the last line into its individual parts of x, y ,z and initialize variables for eastings northings and  
# elevations

split the line so that x, y, and z are separated

store the x value in a variable min\_easting  
store the x value in a variable max\_easting

store second value in a variable min\_northing  
store second value in a variable max\_northing

Store third value in a variable min\_elevation  
store third value in a variable max\_elevation

for each line of coordinates:  
 split the line so that x, y, z are separated

# find the max and min easting  
 for the first value in each line:  
 if that value is < min\_easting:  
 replace it with min\_easting  
 elif that value is > max\_easting:  
 replace with max\_easting

# find the max and min northing  
 for the second value in each line:  
 if that value is < min\_northing:  
 replace it with min\_northing  
 elif that value is > max\_northing:  
 replace it with max\_northing

# find the max and min elevation  
 for the third value in each line:  
 if that value is < min\_elevation:  
 replace it with min\_elevation  
 elif that value is > max\_elevation:  
 replace it with max\_elevation

# find the size of the tile  
the max\_easting – the min\_easting will give the length of the tile and then / 1000 to convert to km  
the max\_northing – the min\_northing will give the height of the tile and then / 1000 to convert to km

# Output the results  
print out all the information to the user with print statements and relevant formatting :)

# Step 4: Test and Debug

I tested and verified my code by downloading some .xyz files from the Open Data Areas Alberta Project database. I used 4 of these files to make sure my code could work with any .xyz files. One of the bugs I ran into was initializing the min/max variables for easting, northings, and elevation. I had to go through the line counting, and then take that last line and split it and use that to initialize all the variables following it. After that I could go through each line in the file, split each line into its components, and then go through with the if/elif workflow to get the min/max values of everything. Another problem was making sure that when I was testing my script, I had to make sure my file input was correct. If I directly pasted the copied path for the file from file explorer into the terminal it did not like it at all. If I removed the quotation marks around it when it gets pasted, then it would work. To get around this user error I decided to use a print statements to clarify the format of the file location specifying that only .xyz files work and that the user should enter in the location using forward slashes as the don’t cause any issues. I think if this script were to go further the biggest thing I could add or change is creating some validation for file input so that the script itself would change any backslashes to forward slashes or double slashes would be changed to single slashes. Its kind of hard to do or implement at the level we are at right now though.