Background (least squared regression):

Least squared regression is a popular method to find the line of best fit. Although I wanted to go over how to do it in class, we don't have time to do it. I'll do my best to explain it through these words and examples on this paper ☹

The goal is to calculate the slope (m) and y-intercept (b) in the equation of the line:

$$y = mx + b$$

The steps to compute the line of best fit for *N* ordered pairs:

1. For each point (x, y), calculate x^2 and xy

x	y	x^2	xy
43.0791753	43.2385799	1855.81535	1862.68236
16.9736206	17.6961419	288.103795	300.367598
34.4956083	34.568358	1189.94699	1192.45654
4.71798562	5.23624523	22.2593884	24.7045297
41.6967421	42.5515545	1738.6183	1774.2612
41.9783665	42.3846592	1762.18325	1779.23876
15.3294726	14.6096058	234.992729	223.957552
14.4867023	15.0703553	209.864545	218.319751
43.658919	44.0158368	1906.10121	1921.68385
19.3269891	19.7741542	373.532509	382.174862
30.7861826	30.104145	947.789042	926.791707
32.1273647	31.787452	1032.16756	1021.24706
29.9712786	29.9799551	898.277543	898.537588
2.7209674	2.38086257	7.40366357	6.47824942
23.3858298	24.0954942	546.897036	563.493127
27.0070658	27.1145851	729.381605	732.285385
47.4511768	46.7983566	2251.61418	2220.63709
24.138327	24.6524038	582.65883	595.067784
5.77633769	5.39855287	33.3660771	31.1838644
6.64073508	6.46490598	44.0993624	42.9317279

2. Find $\sum x, \sum y, \sum x^2, \sum xy$

sum(x)	sum(y)	sum(x^2)	sum(xy)
505.748847	507.922204	16655.073	16718.5006

3. Calculate the slope (N is the number of ordered pairs):

$$m = \frac{N\sum xy - \sum x \sum y}{N\sum x^2 - (\sum x)^2}$$

$$m = \frac{(20)(16718.5006) - (505.748847)(507.922204)}{20(16655.073) - 505.748847^2} = 1.00219$$

4. Calculate the y-intercept:

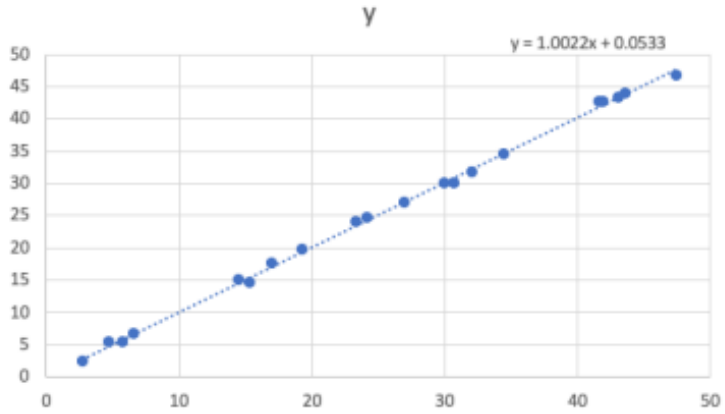
$$b = \frac{\sum y - m\sum x}{N}$$

$$b = \frac{507.922204 - 1.00219(505.748847)}{20} = 0.05327206$$

5. Make our equation $y = mx + b$

$$y = 1.00219x + 0.05327206$$

The graph is shown below (I used Excel not Python). The line of best fit is graphed and so are the points that we used to find the line of best fit.



So, now that you've seen the algebraic method, let's see the linear algebra method!
The setup is based on this matrix equation:

$$y = X\begin{bmatrix} m \\ b \end{bmatrix}$$

y is a nx1 matrix of y-coordinates
X is an nx2 matrix where the first column is the x-coordinate. The second column is 1 for matrix multiplication purposes.

To find the slope (m) and the y-intercept (b), use…

$$\begin{bmatrix} m \\ b \end{bmatrix} = (X^T X)^{-1} X^T y$$

Let's use the same points as last time to find the best fit line with this method.
Note: X (not x) is a matrix and it looks like this:

X	
43.0791753	1
16.9736206	1
34.4956083	1
4.71798562	1
41.6967421	1
41.9783665	1
15.3294726	1
14.4867023	1
43.658919	1
19.3269891	1
30.7861826	1
32.1273647	1
29.9712786	1
2.7209674	1
23.3858298	1
27.0070658	1
47.4511768	1
24.138327	1
5.77633769	1
6.64073508	1

The first column has all of the x's (like in the previous example). The second column is full of 1's. This is for the y-intercept.
The y is the same as in the last example.

The calculations are as follows:

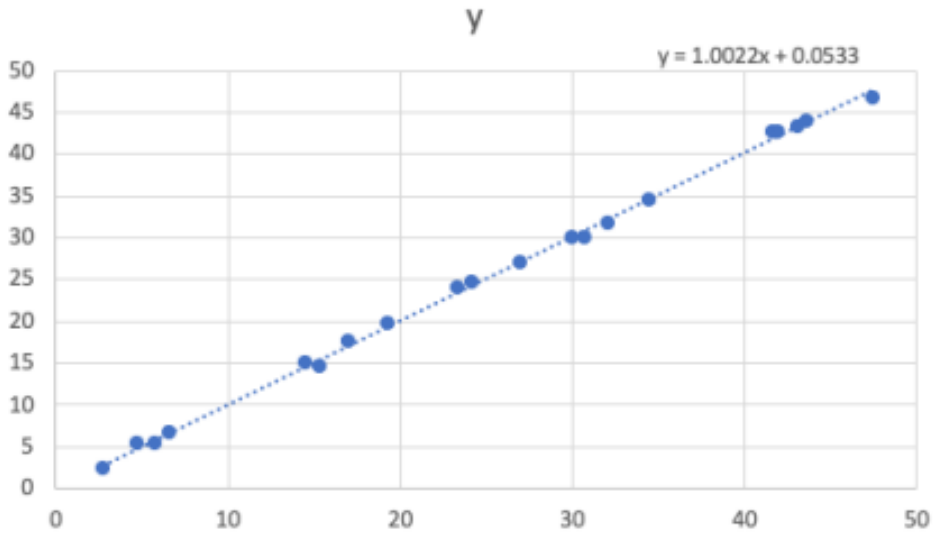
$$X^T X = \begin{pmatrix} 16655 & 505.749 \\ 505.75 & 20 \end{pmatrix}$$

$$(X^T X)^{-1} = \begin{pmatrix} 0.00025867 & -0.006541 \\ -0.006541 & 0.21540568 \end{pmatrix}$$

$$X^T y = \begin{bmatrix} 16718.5006 \\ 507.922204 \end{bmatrix}$$

$$\begin{bmatrix} m \\ b \end{bmatrix} = (X^T X)^{-1} X^T y = \begin{bmatrix} 1.0022 \\ 0.0533 \end{bmatrix}$$

And we get the same results! ☺



Task:

1. Take a close look at the `lin_reg.py` file. There are four empty functions: `least_sq(file_name)` and `mat_least_sq(file_name)` and `predict(file_name, x)` and `plot_reg(file_name, using_matrix)`. Read through all of their descriptions carefully. Remember, you will lose points if you do not follow the instructions. We are using a grading script

Summary of function tasks

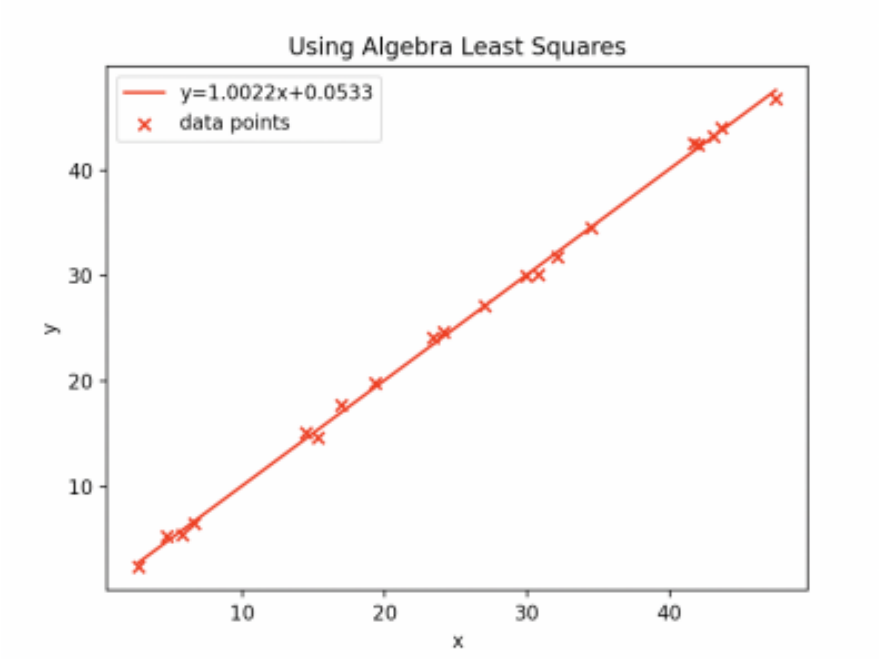
`least_sq(file_name):` Given the csv `file_name`, find the slope and y-intercept of the data using algebraic least squares (the first linear regression presented). You need to return the slope and y-intercept IN THAT ORDER. Round the slope and y-intercept to four decimal places.
`mat_least_sq(file_name):` Given the csv `file_name`, find the slope and y-intercept of the data using linear algebraic least squares using matrices (the second linear regression presented). You need to return the slope and y-intercept IN THAT ORDER. Round the slope and y-intercept to four decimal places.
`predict(file_name, x):` Given the csv `file_name` and an input value `x`, predict what the output would be using the equation that is derived from `mat_least_sq()`. This means that you should be calling `mat_least_sq()` in this function. Round the predicted output to four decimal places before returning the value.
`plot_reg(file_name, using_matrix):` Given the csv `file_name` and an indicator of which linear regression method to use `using_matrix`, output a graph of the data points and the line of best fit. • If `using_matrix=False`, then you should be plotting your results from `least_sq`. You should be using red for everything in the graph with X markers for the data points. • If `using_matrix=True`, then you should be plotting your results from `mat_least_sq`. You can use any color but the default blue and red. You can use any data point marker except for the default dot and X. `plot_reg()` should not return anything. Your graphs should also contain the following: • Labeled x axis • Labeled y axis • Graph Title • Legend (see example for details)

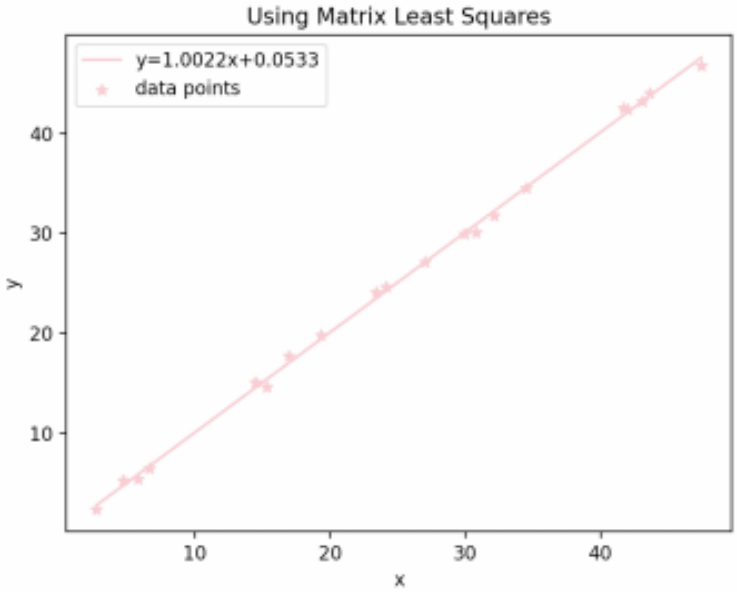
Some important notes:

- For consistency's sake, do not round until the very end. Meaning you should not round anything until you return your answers.
- Hint: to plot the best fit line, find the smallest and largest x-coordinate. Plug these x-coordinates into the linear equation and plot them.
- If you want to create extra functions/methods to assist you, feel free to do so. However, we will only be testing the three functions that are originally in the file.
- **If you use any library's linear regression or least squares method function, you will get an automatic zero. You must implement this on your own!**

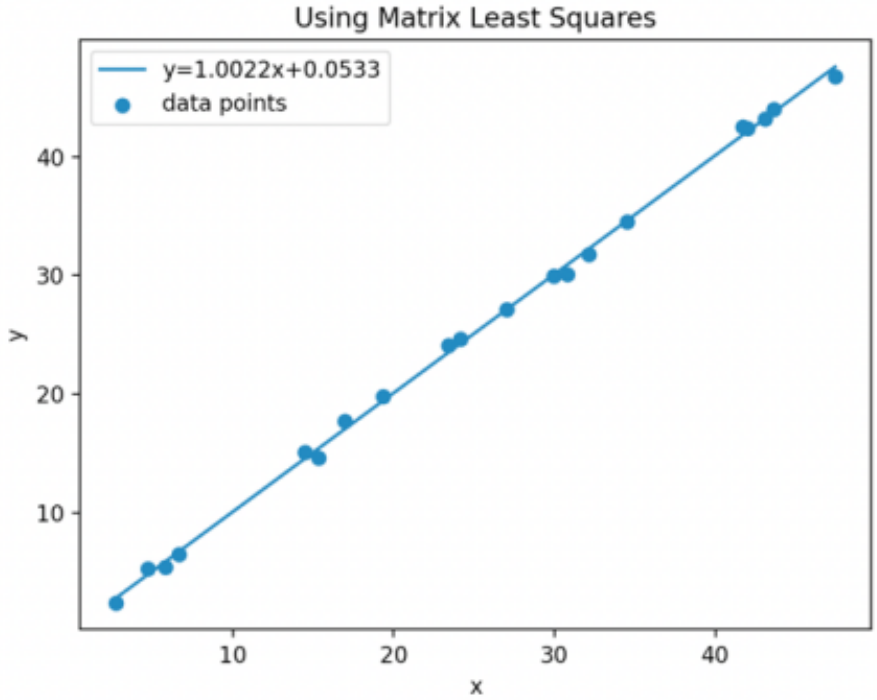
2. Your job is to implement all four of these functions so that it passes all test cases. We provide one csv file for you to test on (`data.csv`), but we will be using other data sets and csv files to check if your work is correct.
3. By running the test case provided (`data.csv`), you should get the following results:

```
Slope using algebraic least squares: 1.0022
y-intercept using algebraic least squares: 0.0533

Slope using linear algebra least squares: 1.0022
y-intercept using linear algebra least squares: 0.0533

Extrapolation: 100.2733
Interpolation: 38.1369
```





Note: your "matrix using least squares" graph may have different colors and markers from mine.



In NO CASE should your graphs have the dot marker or the blue color shown above!

4. If you feel confident in your program so far, run your program after changing the test case's `csv_file` from "`data.csv`" to "`data2.csv`"
5. Take screenshots of the two graphs you obtain (one from using algebraic least squares and the other from matrix least squares). Put these two screenshots in a pdf or word file. You will be submitting this with your py and txt files

Some helpful functions

Function name	What it does
`round(x, y)`	Rounds the value, x, to y decimal places: Example: `round(1.23456, 3) => 1.235`
`matrix_name.T`	Transposes matrix
`np.ones(num)`	Creates a vector full of ones. There will be `num` ones. Example: `np.ones(3) =>` $\begin{bmatrix}1\\1\\1\end{bmatrix}$

`np.column_stack((col1, col2))`	Concatenates two 1d numpy arrays to make a 2d numpy array. If $x = \begin{bmatrix}1\\2\\3\end{bmatrix}$ *and* $b = \begin{bmatrix}1\\1\\1\end{bmatrix}$ `np.column_stack(x,b)` => $\begin{bmatrix}1 & 1\\2 & 1\\3 & 1\end{bmatrix}$
`np.linalg.inv(mat_name)`	Finds the inverse of the matrix `mat_name`