1 | One:

- I would expect to see numbers similar to the y intercept and slope of the line that the model is trying to fit (□=0.3□+1). In this particular example, I would expect to see an intercept close to 1, and a coefficients close to 0.3.

2 | Two:

- I expected it to print out the corresponding y values when plugged back into the original equation.

3 | Three:

- I expected to see a line similar to the graph of □=0.3□+1.

4 | Four

- I changed the equation of the line to `data_one_x['y'] = 1 * data_one_x['x'] + 1` and verified that the code still functioned. The output was `Intercept: [1.] Coefficients: [[file:1..org][1.]]` meaning that it came to the correct answer, verifying that the code was working properly.

5 | One

- I expected it to print numbers similar to the definition of the plane: `y_two_x = 0.5 * x1_two_x - 2.7 * x2_two_x- 2 + noise_two_x` `(0.5, -2.7, -2)`

6 | Two

- I expected to see a plane similar to the one defined above.

7 | Three

- I decided to change the definition of the graph to `y_two_x = 1 * x1_two_x + 1 * x2_two_x +1 + noise_two_x` and see if the code still functioned. `print_model_fit` printed `Intercept: 1.061603912300199 Coefficients: [0.97499882 0.96615802]`, showing that the code was working properly.

8 | Four

- The only major differences were in the visualization section. I would imagine that these visualizations are very helpful with graphs containing few dimensions, but become far less useful as the math stays the same and the dimensions increase.