1. Galileo found that a body undergoing uniform acceleration (such as one freely falling in a vacuum) moves a distance \( d = \frac{1}{2}gt^2 \) in time \( t \). In the absence of air resistance, acceleration \( g \) should equal 9.8 m/sec\(^2\) on the Earth.

Not too long ago I dropped a beanbag off the 2nd through 6th floor balconies down to the ground floor of the Mitchell Institute building. The drop times were 1.18, 1.61, 2.00, 2.23, and 2.48 seconds. The corresponding heights were 5.61, 10.15, 14.69, 19.23, and 23.77 meters.

So, let us consider a dataset that can be fit by a 2nd order fit through the origin, or \( y = ax^2 \).

Here the \( y \)'s are the drop distances, the \( x \)'s are the drop times, and \( a = g/2 \) is the coefficient of the quadratic term.

a. Using Gauss’s method of least squares, show that

\[
a = \frac{\sum_{i=1}^{N} y_i x_i^2}{\sum_{i=1}^{N} x_i^4}.
\]

b. What is the implied value of \( g \) from this dataset? Make a graph of the data and the best fit through the origin.

c. Line 44 of the python program multifit.py is as follows

\[
xx = linspace(x.min(), x.max(), 100) \quad \# \quad \text{[comment here]}
\]

Using the nano editor or any other editor you prefer, change this line to

\[
xx = linspace(0.0, x.max(), 100)
\]

Now run program multifit.py using beanbag.dat as the data file. Do a second order fit of the form \( y = a + bx + cx^2 \). (Nevermind that the uncertainties of the coefficients are not defined. Python’s module polyfit expects there to be a minimum of 6 points for a 2nd order fit.) Notice how the resulting curve does not go through the origin. Here we have the absurd result that one could drop a beanbag from a height of 3.78 m and it would take 0.0 seconds exactly to hit the floor. One must use the right function to fit a dataset and one should not extrapolate very far beyond either end of the dataset!
2. Write what might your first ever program in Python. You could benefit by looking at some of the programs in your python directory in the Mac lab, such as rzeta.py.

Have your program sum up the first N positive integers. First prompt the user to enter the number N. Define an accumulator such as “sum” and initialize it to 0. Start out your loop as follows:

```python
for i in range(N):
```

And remember that Python and C++ start counting from 0, not 1. The program should end by printing out to the user the final value of sum.

Turn in a copy of your few lines of code. What is the sum of the first 10 integers? The first 100?

3. Add some code to the program linearfit.py in your python directory in the Mac lab to double check the extent that the results from polyfit are correct. Your code should explicitly compute the slope and intercept of a linear fit, and the uncertainties of the slope and intercept. You can use as your data file bdiff.dat at your python directory in the Mac lab. It might be useful to look at the program mkhist.py for an example of making a statistical calculation.

You will have to indent your code a minimum of 4 spaces because of how linearfit is written.

Define five accumulators to calculate various sums and initialize them all to 0.0. Call them sx, sy, Sxx, Sxy, and SSres.

We already have the number of points (N) from reading in a dataset and invoking len(x).

Make a little loop to add up the individual x values, and the individual y values. Note that the elements of an array such as x are referred to using square brackets, like x[i] if the index parameter i is being used.

The mean values of x (call it xbar) and the mean value of y (call it ybar) will be the final values of sx and sy divided by the number of points.

Print out the mean X and mean Y values to the screen.

Now we need to calculate a couple more sums in another loop.

$$S_{xx} = \sum_{i=1}^{N} (x_i - \text{xbar})^2.$$
\[ S_{xy} = \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y}) . \]

The least-squares slope will be equal to \( S_{xy}/S_{xx} \). The intercept will be equal to \( \bar{y} - \text{slope} \times \bar{x} \).

Finally, we need to calculate the sum of squares of residuals of the fit:

\[ SS_{res} = \sum_{i=1}^{N} (y_i - (\text{intercept} + \text{slope} \times x_i))^2 . \]

The root-mean-square residual of the fit will be \( s = \text{math.sqrt}(SS_{res}/(N-2)) \). The uncertainty of the intercept will be

\[
\text{siga} = s \times \text{math.sqrt}(1.0/N + \bar{x}\times\bar{x}/S_{xx})
\]

and the uncertainty of the slope will be

\[
\text{sigb} = s / \text{math.sqrt}(S_{xx})
\]

Print out to your terminal the intercept and its uncertainty, the slope and its uncertainty, and the RMS residual.

Hand in your a comparison of the results from polyfit, and your results. Also email a copy of your code to krisciunas@physics.tamu.edu.