Week 7, Lecture 2

Today we continue with reading data from files, except we make it numeric data where we find the mean and variance, as we did in the last example of the last lecture.

To begin, we rewrite the code so that the mean and variance can be computed by making ONE pass through the data instead of the usual TWO. Alongside that we also write the usual mean and variance routines that need two passes, so we can check that the results are correct. The one pass routine is interesting in itself but you don't have to understand the computation.

We put these routines in a separate file that we can later import into another file. So this file becomes a "module", i.e., a library of routines we can use. If you keep adding stat functions to this library, you will have your own stat library in time. At the bottom of this stat module we put a conditional statement that ensures the function test() will run as if it is main() when we run this module directly. But function test() will not run if we import this module into another file that we run directly. This is one of the main uses of the conditional statement using the special variable __name__. So now you know how to build your own Python libraries.

The first couple of programs show you how to import this stat module and call functions inside it. And by running the stat module directly you see how function test() runs as main() to test the functions.

For the next couple of programs we make a new module called “try_stats”. This is exactly the same file as the stat file, but we have added a couple of lines to see how “exceptions” work. For this we need a “try” block of code (i.e., Python runs the code with an eye to “catching” exceptions — which are runtime errors such as
a divide by zero etc.). When a particular exception occurs, Python control moves directly to a portion of the code
which must handle the exception. In our examples we have simple print statements to show that an exception has
occurred. We don’t do anything special to handle the exception, though in general you will have some code in this part which tries to “handle” the exception, aside from printing something. What is done will depend on your application.

The last example shows you (a) how to compute min and max of a sequence of n numbers in 2 separate passes where each costs O(n) comparisons, so the total cost is O(2n). Next you see an algorithm where the list can be divided into pairs of numbers and the min and max comparison done in one pass through the list in a somewhat clever way that reduces the complexity from O(2n) to O(3*n/2) = O(1.5n).

Your homework for the weekend is convert the algorithm into a Python program. It’s simple to do and good practice.

We have now completed Chapters 1 through 8 (with the exception of Chapter 4 on Graphics). We’ll see that soon.