Instructions. Implement the following functions in a file called **hw1.py**, the jupyter notebook in Question 6 in a file called **hw_examples.ipynb**, and submit both of them (i.e., make sure both are pushed) to your GitHub repository for HW-1 by the end of the day on the due date. More details on submitting to GitHub will be provided in Piazza, including where to obtain the starter code. Note that initial versions of both files will be provided as part of the starter code for this homework. To receive full points for this assignment, you must: (a) follow standard python formatting style; (b) include your name, the course, and the semester in a file comment header (for **hw1.py**); and (c) provide comments as appropriate in your function definitions.

1. [5 pts] Write a function called **list_stats** that takes an \( n \)-element list of numeric values and returns the min, max, average, and sum of the values in the list as a 4-tuple. For example, the call **list_stats([1,2,3,4,5])** should return the tuple \((1, 5, 3.0, 15)\), the call **list_stats([1])** should return the tuple \((1, 1, 1.0, 1)\), and the call **list_stats([])** should return the tuple \((None, None, None, None)\).

2. [5 pts] Write a function called **convert_numeric** that takes a string and converts it to an int if the string contains an int, a float if the string contains a float, and the string otherwise. For example, the call **convert_numeric('foo')** should return the string 'foo', the call **convert_numeric('42')** should return the int (not a string) 42, and the call **convert_numeric('3.14')** should return the float (not the string) 3.14.

3. [5 pts] Write two different functions from scratch that create \( m \times n \) matrices (with \( m \) rows and \( n \) columns) as two-dimensional lists (with \( m \) sublists of length \( n \)) containing randomly generated integer values: (a) one called **random_matrix_for** that uses a for loop to create the result; and (b) one called **random_matrix_comp** that uses a list comprehension to create the result. Your functions should use python's **random.randint\((i, j)\)** to generate psuedo-random integer values (from \( i \) to \( j \), inclusive) to fill the lists such that each call should generate a number from 0 up to but not including \( m \times n \). For example, the call **random_matrix_for(2, 3)** might return (depending on the random values generated) \([[3, 0, 2], [1, 2, 3]]\), and similarly for **random_matrix_comp(2, 3)**.

4. [5 pts] Write a function called **transpose_matrix** from scratch that takes an \( m \times n \) matrix (represented as a nested list as in Question 3) and returns a transposed version, i.e., an \( n \times m \) matrix (also represented as a nested list as in Question 3). For example, **transpose_matrix([[1, 2], [3, 4], [5, 6]])** should return **[[1, 3, 5], [2, 4, 6]]**. Note that if we transpose the transpose of a matrix, it should result in the same matrix as the original. Your function should not
modify the given matrix (i.e., it should return a new version of the matrix that is the transpose).

5. [5 pts] Write a function called \texttt{reshape_matrix} from scratch that takes an \(m' \times n'\) matrix (represented as a nested list), a new \(m\) value, and a new \(n\) value, and returns a “reshaped” version of the given matrix such that:

- if \(m \leq m'\) then only the first \(m\) rows of the matrix should be returned
- if \(n \leq n'\) then only the first \(n\) columns of the matrix should be returned
- if \(m > m'\) then all \(m'\) rows of the matrix should be returned with an additional \(m' - m\) rows (after the first \(m\) rows) filled with 0 values for each column
- if \(n > n'\) then all \(n'\) columns of the matrix should be returned with an additional \(n' - n\) columns (after the first \(n'\)) filled with 0 values for each row

For example, the call \texttt{reshape_matrix([[1, 2, 3], [3, 4, 5]]), 3, 2) should return the matrix [[1, 2], [3, 4], [0, 0]]. As in Question 5, your function should not modify the input matrix.

6. [10 pts] Add tests to the jupyter notebook \texttt{hw1_examples.ipynb} (provided with the starter code) for each function above. In addition, be sure to update the notebook with your name at the top (see the notebook for where to put your name). For each function, you should develop enough executable tests to demonstrate that your function works correctly. At the end of the notebook (in Section 8), add text describing any challenges or issues you ran across in this homework assignment. If you didn’t have any issues or challenges, state this in the section.