**Week 7 In-Class Exercises (Extra)**

Part II: Lists

**Q1: Sorting [ \*\*\* ]**

Write a function call sort\_strings().This function takes in a list of strings as its parameter. The function returns a new list of strings that contains all the strings in the original list but in **increasing order of their lengths**. The original list should remain unchanged. For strings of equal length, they should appear in the same order as they are in the original list.

For example, sort\_strings(['abc', 'a', 'xy', '12', 'x']) should return the list ['a', 'x', 'xy', '12', 'abc'].

**Hint:**

* You can gradually add strings in the original list to the new list, but you need to be careful about where to add each string to the new list.
* You may need to insert a string into the new list. Suppose you want to insert string my\_str into the 4th position of my\_list, you can do the following:

my\_list = my\_list[0:4] + [my\_str] + my\_list[4:]

Alternatively, you can use a method called insert() for lists. See <https://docs.python.org/3.6/library/stdtypes.html#mutable-sequence-types> and look for s.insert(i, x). **(Note that insert() is out of scope.)**

**Q2: Matrix Transpose and Matrix Multiplication [ \*\*\* ]**

A matrix can be represented as a list of lists of numbers.

For example, given the following matrix:

We can use the following list to represent it:

[[1.0, 2.0, 1.5], [2.5, 3.0, 2.0], [4.5, 1.5, 2.5]]

You can see that each element of the list above corresponds to a single row of the matrix.

**Part (a)**

The transpose of a matrix changes its row vectors into column vectors. For example, the transpose of the matrix above is the following:

Define a function call get\_matrix\_transpose() that takes in a list that represents a matrix as its parameter. The function returns a new list that represents the transpose of the original matrix.

Note that the input matrix can be of any dimension , where may be different from .

For example:

* get\_matrix\_transpose([[1.0, 2.0, 1.5], [2.5, 3.0, 2.0], [4.5, 1.5, 2.5]]) should return [ [1.0, 2.5, 4.5], [2.0, 3.0, 1.5], [1.5, 2.0, 2.5] ].
* get\_matrix\_transpose([[1.0, 2.0], [2.5, 3.0], [4.5, 1.5]]) should return [ [1.0, 2.5, 4.5], [2.0, 3.0, 1.5] ].

**Part (b)**

An matrix can be multiplied by an matrix. See the definition of matrix multiplication below:

<https://en.wikipedia.org/wiki/Matrix_multiplication#Definition>

Define a function called multiply\_matrices(). The function takes in two lists, each representing a matrix. You can assume that the number of columns of the first matrix is always equal to the number of rows of the second matrix. (Otherwise it’s not possible to multiply the two matrices.) The function returns the result of matrix multiplication

For example,

So multiply\_matrices( [[1.0, 2.0, -1.0], [2.5, 1.5, 0.5]], [[2.0, 2.0], [1.5, 1.0], [0.5, 2.0]] ) should return [[4.5, 2.0], [7.5, 7.5]].

**Hint:**

* You might want to take the transpose of the second matrix first.
* You might want to introduce a function that calculates the dot product (<https://en.wikipedia.org/wiki/Dot_product>) of two vectors.

**Q3: Flattening a List of Numbers [ \*\*\* ]**

Implement a function called flatten(). The function takes in a list called my\_list as its parameter, which contains numbers and possibly lists as its elements. These lists also contain numbers and possibly lists as their elements. This can go on recursively.

For example, my\_list may be

[3, [4, 8, 0], 5]

or

[4, 39, [35, 12, [45], 32], 4, [35, [4, [6]]]]

The function returns a list of numbers that is flattened from the original list.

For example, flatten([4, 39, [35, 12, [45], 32], 4, [35, [4, [6]]]]) returns [4, 39, 35, 12, 45, 32, 4, 35, 4, 6].

**Hint:**

* To check if a variable if of a certain date type, you can use the isinstance() function. See <https://docs.python.org/3.6/library/functions.html#isinstance> .

For example, to check if the variable x is a list, you can use isinstance(x, list), which will return either True or False.

* You can write a ***recursive function*** to solve the problem.