**Lab 3: Libraries and modules**

**Problem 1 –** Import the Matplotlib library and perform the following exercises:

1. Create a 1-D list of integers *X = (-5, 5).* Create two new lists X1 = 3\*X and X2 = X2 and plot X1 and X2 against X on the same plot.
2. Put the appropriate label names and legend on the plot.
3. The line representing X1 show be in red color while the line representing X2 should a dotted blue line. Insert grid lines in the plot.
4. Determine the intersection points of the two lines computationally and mark them with a black circle marker.

**Problem 2 –** Import the NumPy library and perform the following exercises:

1. Create a 2-D array (i.e., matrix) of size (4 x 4), where the value of every element in row *i* and column *j* is *i-j.* Print the array.
2. In the array replace all negative numbers with their squares and the positive numbers with their square roots and print the array.
3. Split the array into two equal size arrays and add the arrays.
4. Multiply the two arrays, making use of a transpose operation if required.

**Problem 3 –** Using a module to modularize code (Do not use NumPy for the given problem but use Matplotlib)

1. Create a module that contains the following functions that act on 2-D lists: (i) Mean (ii) Standard Deviation (iii) Transpose
2. Read the provided file: Pattern.txt, which is a file of integer numbers and convert it to a list of integer numbers {Note: make sure after you read the file that you eliminate any non-integer characters before converting to a list of integers}
3. Reshape the file into a 2-D list of 997 rows and 1000 columns.
4. Plot the 2-D array using imshow() just to get an idea of what the data looks like
5. Import the custom module created by you and perform the operations on the 2-D list (i.e., find the means/ Std-dev/ transpose). Plot the returned values (means etc).
6. Repeat steps (c-e) using inbuilt functions of NumPy, by converting the 2-D list to a NumPy array.

**Bonus**: Read up on the time module. You can import the time module (import time) for Problem – 3, which allows you to compare the time it takes to execute the operations using NumPy and using regular 2-D lists without using NumPy.