# 311 – Numerical Computations
# Lab 7: Numpy Arrays/2D Arrays & Solving Systems of Linear Equations

## PreLab) List comprehension in Python:

```
a=[ x*2 for x in range(4)]
print(a)                                  #[0, 2, 4, 6]
b=[ x+1 for x in a]
print(b)                                  #[1, 3, 5, 7]
c=[x**2 for x in a if x %4 ==0]      #[0,16]
```

**so:**

```
L= input("Enter integers").split( )
L= [ int(x) for x in L]
print (L)
```

**will convert a list of strings to list of integers**

**A) The NumPy arrays are homogeneous, this makes any operation on array elements very fast.**

```
import numpy as np
ar = np.array([13,13,'a']) print(ar)

----------
Output:
['13' '13' 'a']
=====================================
import numpy as np

b = np.array([13, 13, 3.5])
print(b)

----------
Output:
[13.  13.   3.5]
====================================
```

## B) Two Dimensional arrays in Numpy

```
import numpy as np

ar = np.array([[9,2,-5], [3,4,-2], [-9,5,-3]])

print(ar)
-----------
Output:
[[ 9  2 -5]
 [ 3  4 -2]
 [-9  5 -3]]


================================
```

- **Slicing matrices in Numpy:**

```
import numpy as np

arr = np.array([[1, 2, 3, 4], [9, 7, 6, 2], [4, 5, 2, 1]])

print(arr[0:2, 1:4])

output:
[[2 3 4]
 [7 6 2]]
=====================================
```

## Some Numpy function that we will need:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
print(arr)
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr)
print(arr[0])
a = np.zeros(3)
print(a)
b = np.ones((2,4))
print(b)
print(b.shape)
print(b.shape[0])
print(b.shape[1])
print(b.size)
c = np.full((3,3),7)
print(c)
```

```
[1 2 3 4 5]

[[1 2 3]
 [4 5 6]]

[1 2 3]

[0. 0. 0.]

[[1. 1. 1. 1.]
 [1. 1. 1. 1.]]

(2, 4)
2
4
8

[[7 7 7]
 [7 7 7]
 [7 7 7]]
```

**Remark: In Python,**

```
L = [0]* 5
print (L)
```

**will print: [0, 0, 0, 0, 0]**

## A)How to read a (square) matrix in Numpy:

```
L = input("Enter matrix A elements: ").split( )
L = [float(x) for x in L]
n = int(sqrt(len(L)))    #or you can ask for n from user
A = np.array(L).reshape(n, n)
print(A[0][0])      #print an element from matrix
```

**Q1:** Using Numpy: write a program that reads a matrix (read its dimension at start) and prints the sum of each row and each column.

**Example:**

```
Enter number of rows: 2
Enter number of columns 3
Enter 6 matrix elements:  3 5 2 1 8 4

Sum of row 0= 10
Sum of row 1= 13
Sum of column 0= 4
Sum of column 1= 13
Sum of column 2= 6
```

## D) Operations on Matrices in Numpy:

```
import numpy as np
A = np.array(([3,2],[5,4]))
B = np.array(([5,8],[6,2]))
S = A +B            # also you can use  S=A-B
print(S)
S = A + 5             #adds scalar 5 to all elements of A
P=np.dot(A,B)         #multiplication
T=P.transpose( )     # transpose
```

# E) Solving a linear system:

```
import numpy as np

a = np.array([[1,2,-3], [2,-5,4], [5,4,-1]])
b = np.array([-3,13,5])
x = np.linalg.solve(a, b)

print(a)
print(b)
print(x)

print(np.allclose(np.dot(a, x), b))
```

```
[[ 1  2 -3]
 [ 2 -5  4]
 [ 5  4 -1]]

[-3 13  5]

[ 2. -1.  1.]
True
```

## F) !!! Singular Matrix!!!

```
import numpy as np

a = np.array([[2,4], [3,6]])
b = np.array([5,17])
x = np.linalg.solve(a, b)

print(x)   # !!!! Singular Matrix
```

**How to protect against that:**

```
if (np.linalg.det(a)==0):
        print("Singular")
else:
       print("Not Singular")
```

=====================================

**Q2) Write a python program that reads the elements of a square matrix A (assume a perfect square number of inputs), and the b vector (assume correct size), and solve the system: Ax=b.**

**If the matrix is singular, print “No Unique Solution”.**
**Otherwise print the solution: x.**

**Assume no error in number of inputs.**