

# Assignment 2:

Name: Krish Jain

Roll No: J069

1. Prove properties of matrix multiplication
2. Write notebook in a structured manner
3. Calculate inverse of a matrix using numpy (inbuilt api and/or manual coding)
4. Show how numpy is faster than traditional looping
  - A. You have to print time for both cases
  - B. Use a large sized matrix (10000 x 10000) or something even larger. You can use any example.

In [ ]:

```
import numpy as np
```

In [11]:

```
a = np.random.randint(1,20,9)
a = a.reshape((3,3))
print(a)
```

```
[[ 3 18 14]
 [16  4 18]
 [ 6 12  8]]
```

In [13]:

```
b = np.random.randint(1,20,9)
b = b.reshape((3,3))
print(b)
```

```
[[ 7 18 17]
 [ 7 13  1]
 [ 2 13 16]]
```

In [15]:

```
c = np.random.randint(1,20,9)
c = c.reshape((3,3))
print(c)
```

```
[[ 4  4  2]
 [ 4 18 17]
 [ 7  9 11]]
```

## Commutative property

**A.B=B.A**

In [16]:

```
a.dot(b)
```

Out[16]:

```
array([[175, 470, 293],
       [176, 574, 564],
       [142, 368, 242]])
```

In [17]:

```
b.dot(a)
```

Out[17]:

```
Out[17]:  
array([[411, 402, 558],  
       [235, 190, 340],  
       [310, 280, 390]])
```

In [18]:

```
print(a.dot(b)==b.dot(a))
```

```
[[False False False]  
 [False False False]  
 [False False False]]
```

In [19]:

```
#Matrices are non-commutative
```

## Associative property

In [26]:

```
x = np.matmul(a,np.matmul(b,c))
```

In [27]:

```
y = np.matmul(np.matmul(a,b),c)
```

In [28]:

```
print(x==y)
```

```
[[ True  True  True]  
 [ True  True  True]  
 [ True  True  True]]
```

In [29]:

```
#Matrices are associative
```

## Distributive property

In [31]:

```
x = np.matmul(a,(b+c))
```

In [32]:

```
y = np.matmul(a,b)+np.matmul(a,c)
```

In [33]:

```
print(x==y)
```

```
[[ True  True  True]  
 [ True  True  True]  
 [ True  True  True]]
```

In [34]:

```
#Matrices follow the distributive property
```

## Multiplicative Identity

In [42]:

```
I = np.identity(3)
```

```
I
```

```
Out[42]:
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
In [45]:
```

```
x = np.dot(a,I)
y = np.dot(I,a)
print(x==y)
```

```
[[ True  True  True]
 [ True  True  True]
 [ True  True  True]]
```

## Multiplicative property of zero

```
In [46]:
```

```
zero = np.zeros(9).reshape(3,3)
```

```
In [47]:
```

```
print(np.dot(a,zero))
print(np.dot(zero,a))
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

## Matrix Inverse

```
In [48]:
```

```
print(a)
```

```
[[ 3 18 14]
 [16  4 18]
 [ 6 12  8]]
```

```
In [50]:
```

```
print('inverse = \n', np.linalg.inv(a))
```

```
inverse =
[[-0.12777778  0.01666667  0.18611111]
 [-0.01388889 -0.04166667  0.11805556]
 [ 0.11666667  0.05       -0.19166667]]
```

## Numpy vs looping

```
In [ ]:
```

```
import time
size = 10000
```

```
In [1]:
```

```
A = np.random.randn(size,size)
B = np.random.randn(size,size)
listA = [list(i) for i in A]
```

```
listB = [list(i) for i in B]
startloop = time.time()
listC = []
for i in range(size):
    row = []
    for j in range(size):
        row.append(listA[i][j]+listB[i][j])
    listC.append(row)
endloop = time.time()
print('time taken for loop:', endloop-startloop)
```

time taken for loop: 163.8115156411

In [2]:

```
startnp = time.time()
npC = np.
endnp = time.time()
print('time take for numpy:',endnp-startnp)
```

time take for numpy: 20.28611686813