In [1]:

```
import numpy as np
np.random.seed(1340)
A = np.random.randn(3, 3)
B = np.random.rand(3, 3)
C= np.random.rand(3, 3)
I = np.identity(3)
```

In [3]:

```
print('Matrix A : \n', A)
print('Matrix B : \n', B)
print('Matrix C : \n', C)
print('Identity Matrix : \n', I)
```

```
Matrix A :
 [[ 0.26146449 -0.94392614  1.18802285]
 [-0.97778601 -0.91616748  0.62434481]
 [ 0.49156832 -0.46211821  0.0840959 ]]
Matrix B :
 [[0.27619423 0.36435099 0.01095459]
 [0.12387259 0.42866279 0.55609389]
 [0.27347351 0.85163987 0.6800704 ]]
Matrix C :
 [[0.69339397 0.88226381 0.42075066]
 [0.6330561  0.70857565 0.77727631]
 [0.7521562  0.61525722 0.2282028 ]]
Identity Matrix :
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [4]:

```
# communatitive property not applicable
AdotB = A.dot(B)
BdotA = B.dot(A)

print('A.B : \n', AdotB)
print('B.A : \n', BdotA)
```

```
A.B :
 [[ 0.28018119  0.70240646  0.28589185]
 [-0.21280512 -0.21726728 -0.09558796]
 [ 0.10152256  0.05262995 -0.19440505]]
B.A :
 [[-0.27865739 -0.5995758   0.55652694]
 [-0.11339406 -0.76663459  0.46156206]
 [-0.42691687 -1.35265647  0.91380084]]
```

In [6]:

```
# Associative property [(A.B).C = A.(B.C)]
AB_C = np.dot(A, B).dot(C)
A_BC = A.dot(np.dot(B, C))

print('(A.B).C : \n', AB_C)
print('A.(B.C) : \n', A_BC)
```

```
(A.B).C :
 [[ 0.85397397  0.92079887  0.72909164]
 [-0.35699724 -0.40051175 -0.28022805]
 [-0.04251013  0.00725286  0.03925992]]
A.(B.C) :
 [[ 0.85397397  0.92079887  0.72909164]
 [-0.35699724 -0.40051175 -0.28022805]
 [-0.04251013  0.00725286  0.03925992]]
```

In [7]:

```
# Distributive property [A.(B+C) = ]
lhs = np.dot(A, B+C)
rhs = np.dot(A, B) + np.dot(A, C)
print("A.(B+C) : \n", lhs)
print('A.B + A.C : \n', rhs)
```

```
A.(B+C) :
 [[ 0.75749964  0.99518367 -0.06667808]
 [-1.00117665 -1.34497382 -1.07663013]
 [ 0.21307956  0.21061779 -0.32757997]]
A.B + A.C :
 [[ 0.75749964  0.99518367 -0.06667808]
 [-1.00117665 -1.34497382 -1.07663013]
 [ 0.21307956  0.21061779 -0.32757997]]
```

In [8]:

```
# Identity property [A.I = I.A]
AI = np.dot(A, I)
IA = np.dot(I, A)

print('A.I : \n', AI)
print('I.A :\n', IA)
```

```
A.I :
 [[ 0.26146449 -0.94392614  1.18802285]
 [-0.97778601 -0.91616748  0.62434481]
 [ 0.49156832 -0.46211821  0.0840959 ]]
I.A :
 [[ 0.26146449 -0.94392614  1.18802285]
 [-0.97778601 -0.91616748  0.62434481]
 [ 0.49156832 -0.46211821  0.0840959 ]]
```

In [10]:

```
# Multiplicative property of zero [A.0 = 0.A = 0]
z_mat = np.zeros(9).reshape(3, 3)
lhs = np.dot(A, z_mat)
rhs = np.dot(z_mat, A)

print('A.0 : \n', lhs)
print('0.A : \n', rhs)
```

```
A.0 :
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
0.A :
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

In [12]:

```
# dimensions on matrix multiplication
m,n,k = 5,7,3
mat_m_n = np.random.randn(m, n)
mat_n_k = np.random.randn(n, k)
mat_mult = np.dot(mat_m_n, mat_n_k)
result_x, result_y = mat_mult.shape
print(f' {m}x{n} matrix X {n}x{k} matrix = {result_x}x{result_y} matrix')
```

```
 5x7 matrix X 7x3 matrix = 5x3 matrix
```

In [13]:

```
# inverse of a matrix
A_inv=np.linalg.inv(A)
A_inv
```

Out[13]:

```
array([[ 0.27832064, -0.61807157,  0.65685131],
       [ 0.51213839, -0.7396516 , -1.74365791],
       [ 1.18739286, -0.45165185, -1.52996012]])
```

In [14]:

```python
# comparison of time between numpy and loops
import time
size = 5000
numpy_mat_A = np.random.randn(size, size)
numpy_mat_B = np.random.randn(size, size)
list_mat_A = [list(i) for i in numpy_mat_A]
list_mat_B = [list(i)for i in numpy_mat_B]
```

In [15]:

```python
start_loop = time.time()
list_mat_C = []
for i in range(size) :
    row = []
    for j in range(size) :
        row.append(list_mat_A[i][j] + list_mat_B[i][j])
    list_mat_C.append(row)
end_loop = time.time()
```

In [16]:

```python
start_numpy = time.time()
numpy_mat_C = numpy_mat_A + numpy_mat_B
end_numpy = time.time()
```

In [19]:

```python
print('Time for loops : ', end_loop - start_loop)
print('Time for numpy : ', end_numpy - start_numpy)
```

```
Time for loops :  6.918421983718872
Time for numpy :  0.051860809326171875
```

In [ ]:

```python
: #numpy is much faster
```