In [43]:
```python
import numpy as np
from time import process_time
```

Proving matrix multiplication properties

In [3]:
```python
#Matrices

a=np.array([[1,2],[3,7]])
b=np.array([[-1,10],[3,1]])
c=np.array([[1,-2],[3,5]])
```

Commutive property

In [35]:
```python
## AB!=BA

print(f'A.B: \n\n{np.matmul(a,b)} \n')

print(f'B.A: \n\n{np.matmul(b,a)}')
```

```
A.B:

[[ 5 12]
 [18 37]]

B.A:

[[29 68]
 [ 6 13]]
```

Associative property of multiplication

In [44]:
```python
## A(BC)= (AB)C

print(f'A(BC): \n\n {np.matmul(a,np.matmul(b,c))} \n')

print(f'(AB)C): \n\n {np.matmul(np.matmul(a,b),c)}')
```

```
A(BC):

 [[ 41  50]
 [129 149]]

(AB)C):

 [[ 41  50]
 [129 149]]
```

Distributive properties

In [11]:
```python
#A(B+C) = AB + AC

print(f'A(B+C): \n {np.matmul(a, b+c)}')

print(f'AB + AC: \n {np.matmul(a,b) + np.matmul(a,c)}')
```

```
A(B+C):
 [[12 20]
 [42 66]]
AB + AC:
 [[12 20]
 [42 66]]
```

Multiplicative identity property

In [47]:
```python
## IA = A and AI=A

print(f'A \n{a}')
print(f'I.A \n {np.matmul(np.identity(2),a)}')
```

```
A
[[1 2]
 [3 7]]
I.A=A:
 [[1. 2.]
 [3. 7.]]
```

Multiplicative property of zero

In [6]:
```python
## 0A= 0 and A0=0

print(f'A0= 0:  \n {np.matmul(a,np.zeros_like(a))}')
```

```
A0= 0:
 [[0 0]
 [0 0]]
```

Dimension property

In [7]:
```python
#mxn and nxp = mxp

print(f'mxn and nxp: \n {np.matmul(a,b)}')
```

```
mxn and nxp:
 [[ 5 12]
 [18 37]]
```

Calculating inverse of a matrix

matrix definition

In [20]:
```python
a= np.array([[8,3,1],[0,1,3],[9,1,0]])
a
```

Out[20]:
```
array([[8, 3, 1],
       [0, 1, 3],
       [9, 1, 0]])
```

calculating the inverse

In [30]:
```python
np.matmul(np.linalg.inv(a), a)
```

Out[30]:
```
array([[ 1.00000000e+00,  0.00000000e+00, -3.46944695e-18],
       [ 0.00000000e+00,  1.00000000e+00,  0.00000000e+00],
       [ 1.66533454e-16,  5.55111512e-17,  1.00000000e+00]])
```

Show how numpy is faster than traditional loop

We will use the example of dot product to see how fast numpy is. First the traditional list

In [ ]:
```python
l1= [i for i in range(10000)]
l2= [i for i in range(10000)]

start=process_time()

dot= 0

for i,j in zip(l1,l2):
    dot += i*j
end=process_time()

print(end-start)
```

In [ ]:
```python
arr1= np.array([i for i in range(10000)])
arr2= np.array([i for i in range(10000)])

start=process_time()

print(np.dot(arr1,arr2))

end= process_time()
print(end-start)
```