Show how numpy is faster than traditional looping

In [16]:
```
import time
import numpy as np
t1 = time.time()
X = range(1000000)
Y = range(1000000)
Z = [X[i] + Y[i] for i in range(len(X)) ]
time1 = time.time() - t1
print(time1)
```

```
0.5006365776062012
```

In [17]:
```
t1 = time.time()
a = np.arange(1000000)
b = np.arange(1000000)
Z = a+b
time2 = time.time() - t1
print(time2)
```

```
0.027371883392333984
```

In [19]:
```
rate = np.divide(time1 , time2)
print(rate)
```

```
18.29017647161298
```

Inverse Of A Matrix

In [22]:
```
mat = np.array([[4 , 5 , 8] , [2 , 1 , 4] , [-4 , 8 , 2]])
print(mat)
```

```
[[ 4  5  8]
 [ 2  1  4]
 [-4  8  2]]
```

In [23]:
```
inv_mat = np.linalg.inv(mat)
print(inv_mat)
```

```
[[ 5.00000000e-01 -9.00000000e-01 -2.00000000e-01]
 [ 3.33333333e-01 -6.66666667e-01  2.56205313e-17]
 [-3.33333333e-01  8.66666667e-01  1.00000000e-01]]
```

Properties Of Matrix Multiplication

Commutative Law

In [25]:
```
a = np.array([[4 , 5 , 8] , [2 , 1 , 4] , [-4 , -3 , 2]])
b = np.array([[1 , 7 , 4] , [3 , 5 , -4] , [-4 , -8 , 6]])
c = np.array([[-1 , 3 , 9] , [-3 , 7 , -6] , [-9 , 7 , 2]])

print(a @ b)
print('--------------------------------------------------------------------
--------------------------------')
print(b @ a)
```

```
[[-13 -11  44]
 [-11 -13  28]
 [-21 -59   8]]
------------------------------------------------------------------------
---------------------------
[[  2   0  44]
 [ 38  32  36]
 [-56 -46 -52]]
```

As both the matrices are different a*b is not equalto b*a

So matrix multiplication is not commutative

Associative Law

In [26]:
```
### Associative Law -----> A(BC) = (AB)C


print(a @ (b @ c))
print('--------------------------------------------------------------------
--------------------------------')
print((a @ b) @ c)
```

```
[[-350  192   37]
 [-202   72   35]
 [ 126 -420  181]]
------------------------------------------------------------------------
---------------------------
[[-350  192   37]
 [-202   72   35]
 [ 126 -420  181]]
```

As both the matrices are same A(BC) = (AB)C

So matrix multiplication is associative

Distributive Law

In [29]:
```
### Distributive Law -----> A(B+C) = AB + AC
### (A+B)C = AC + BC

print(a @ (b + c))
print('--------------------------------------------------------------------
--------------------------------')
print((a @ b) + (a @ c))
```

```
[[-104   92   66]
 [ -52   28   48]
 [ -26  -78   -6]]
------------------------------------------------------------------------
---------------------------
[[-104   92   66]
 [ -52   28   48]
 [ -26  -78   -6]]
```

Hence matric multiplication follows distributive law

Multiplicative Identity

In [30]:
```
I = np.array([[1,0,0] , [0,1,0] , [0,0,1]])

print(a @ I)
print('--------------------------------------------------------------------
--------------------------------')
print(I @ a)
print('--------------------------------------------------------------------
--------------------------------')
print(a)
```

```
[[ 4  5  8]
 [ 2  1  4]
 [-4 -3  2]]
------------------------------------------------------------------------
---------------------------
[[ 4  5  8]
 [ 2  1  4]
 [-4 -3  2]]
------------------------------------------------------------------------
---------------------------
[[ 4  5  8]
 [ 2  1  4]
 [-4 -3  2]]
```

In [ ]:
```

```