Assignment 2

Aryaman Gautam

J001

In [1]:
```
import numpy as np
```

Matrix Addition & Subtraction

In [2]:
```
matrix1 = np.array([[1,2],[3,4]])
matrix2 = np.array([[5,6],[7,8]])

print("matrix1 \n",matrix1,"\n\n\n\nmatrix2 \n",matrix2)
print("\n\n\n\n\n")
print("matrix1 + matrix2 \n",matrix1+matrix2)
print("\n\n\n\n\n")
print("matrix1 - matrix2 \n",matrix1-matrix2)
```

```
matrix1
 [[1 2]
```

```
 [3 4]]


matrix2
 [[5 6]
 [7 8]]




matrix1 + matrix2
 [[ 6  8]
 [10 12]]




matrix1 - matrix2
 [[-4 -4]
 [-4 -4]]
```

Scalar multiplication & (addition , subtraction)

In [3]:

```
print("matrix1 \n",matrix1)
print("\n\n\n\n\n")


print("matrix1 + 5\n",matrix1 + 5)
print("\n\n\n\n\n")
print("matrix1 - 5\n",matrix1 - 5)
print("\n\n\n\n\n")
print("matrix1 * 5\n",matrix1 * 5)
```

```
matrix1
 [[1 2]
 [3 4]]
```

```
matrix1 + 5
 [[6 7]
 [8 9]]
```

```
matrix1 - 5
 [[-4 -3]
 [-2 -1]]
```

```
matrix1 * 5
 [[ 5 10]
 [15 20]]
```

Matrix Multiplication

In [4]:

```
print("matrix1 \n",matrix1,"\n\n\n\nmatrix2 \n",matrix2)
print("\n\n\n\n\n")
print("matrix1 * matrix2\n",np.dot(matrix1,matrix2))
```

```
matrix1
 [[1 2]
 [3 4]]
```

```
matrix2
 [[5 6]
 [7 8]]
```

```
matrix1 * matrix2
 [[19 22]
 [43 50]]
```

In [5]:

```
a = np.random.randint(2,50, size=(5,2))
b = np.random.randint(2,50, size=(2,3))

print("matrix A \n",a,"\n\n\n\nmatrix B \n",b)
print("\n\n\n\n\n")
print("A * B\n",np.dot(a,b))
```

```
matrix A
 [[15 30]
 [20 40]
 [49 23]
 [27  3]
 [10 40]]



matrix B
 [[16 42 36]
 [28  5 38]]




A * B
 [[1080  780 1680]
 [1440 1040 2240]
 [1428 2173 2638]
 [ 516 1149 1086]
 [1280  620 1880]]
```

Properties of matrix

AB =/= BA

i.e non cumulative

In [8]:

```
a = np.random.randint(1,50, size=(5,5))
b = np.random.randint(1,50, size=(5,5))

print("A\n",a)
print("\n\n\n\n")
print("B\n",b)
print("\n\n\n\n")

print("AB",np.dot(a,b))
print("\n\n\n\n")
print("BA",np.dot(b,a))
print("\n\n\n\n")
print("we can see they're not equal")
```

```
A
 [[12 32 22 28 40]
 [33 29  7  5 14]
 [49 20 39 23 33]
 [40 40  6 43 41]
 [39 30 46 16 49]]



B
 [[13 42 15 49 46]
 [38 37 26 19 39]
 [44 40  2 30  2]
 [12  9 14 45 29]
 [48 22 12 17 31]]



AB [[4596 3700 1928 3796 3896]
 [2571 3092 1501 2841 3242]
 [4973 5291 2051 5547 4802]
 [4788 4689 2746 5532 5930]
 [6215 5810 2269 5414 5039]]
```

```
BA [[6031 5274 3575 3762 5866]
 [5232 4739 4017 3288 5586]
 [3224 3868 1598 2800 3714]
 [4058 3595 2477 3102 4334]
 [3779 4024 3206 2957 4840]]
```

```
we can see they're not equal
```

A(BC) = (AB)C

i.e assosciative

In [10]:

```
a = np.random.randint(1,50, size=(3,3))
b = np.random.randint(1,50, size=(3,3))
c = np.random.randint(1,50, size=(3,3))

print("A\n",a)
print("\n\n\n\n")
print("B\n",b)
print("\n\n\n\n")
print("C\n",c)
print("\n\n\n\n")



print("A(BC)")
print(np.dot(a,np.dot(b,c)))

print("\n\n\n\n")

print("(AB)C")
print(np.dot(np.dot(a,b),c))
```

```
print("\n\n\n")
print("The values are the same")
```

```
A
 [[24 28 32]
 [17 38 35]
 [16 23 31]]



B
 [[43 37 25]
 [30 17 33]
 [34  6  1]]



C
 [[39 25  4]
 [32 22 21]
 [13 49 33]]



A(BC)
[[185460 184476  95864]
 [189181 193181  99991]
 [147726 144828  73547]]



(AB)C
[[185460 184476  95864]
 [189181 193181  99991]
 [147726 144828  73547]]
```

```
The values are the same
```

A(B+C) = AB + AC

i.e distributive

In [11]:

```
a = np.random.randint(1,50, size=(3,3))
b = np.random.randint(1,50, size=(3,3))
c = np.random.randint(1,50, size=(3,3))

print("A\n",a)
print("\n\n\n\n")
print("B\n",b)
print("\n\n\n\n")
print("C\n",c)
print("\n\n\n\n")



print("A(B+C)")
print(np.dot(a,(b+c)))
print("\n\n\n\n")
print("AB + AC)")
print(np.dot(a,b) + np.dot(a,c))


print("\n\n\n")
print("They're the same")
```

```
A
 [[14 40 35]
 [39 27 10]
 [32 31 37]]
```

```
B
 [[ 2 16 41]
 [44  6  9]
 [36 37 41]]
```

```
C
 [[ 6 27 26]
 [15 31 19]
 [45 41 13]]
```

```
A(B+C)
[[5307 4812 3948]
 [2715 3456 3909]
 [5082 5409 5010]]
```

```
AB + AC)
[[5307 4812 3948]
 [2715 3456 3909]
 [5082 5409 5010]]
```

```
They're the same
```

Identity matrix

AI = IA = A

In [13]:

```
A = np.random.randint(1,50, size=(3,3))
I = np.identity(3)
```

```
print("A \n {} \n\n\n".format(A))
print("I \n {} \n\n\n".format(I))
print(print("AI \n {} \n\n\n".format(np.dot(A,I))))
print(print("IA \n {} \n\n\n".format(np.dot(I,A))))
```

```
A
 [[48 48 35]
 [17 46 49]
 [32 23 15]]



I
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]



AI
 [[48. 48. 35.]
 [17. 46. 49.]
 [32. 23. 15.]]



None
IA
 [[48. 48. 35.]
 [17. 46. 49.]
 [32. 23. 15.]]



None
```

Inverse matrix

A * A(inv) = I

In [22]:

```
A = np.random.randint(1,50, size=(3,3))
```

```
Ainv = np.linalg.inv(A)



print("A \n {} \n\n\n".format(A))
print("A(inv) \n {} \n\n\n".format(Ainv))


print("A * A(inv) \n",np.dot(A,Ainv))
```

```
A
 [[18 40 13]
 [25  6 11]
 [43 36 14]]



A(inv)
 [[-0.04078431 -0.01202614  0.04732026]
 [ 0.01607843 -0.04013072  0.01660131]
 [ 0.08392157  0.14013072 -0.11660131]]



A * A(inv)
 [[ 1.00000000e+00  2.77555756e-17  1.66533454e-16]
 [ 5.55111512e-17  1.00000000e+00  5.55111512e-17]
 [ 1.11022302e-16 -5.55111512e-17  1.00000000e+00]]
```

comparing time taken to execute via numpy & for loop

In [25]:

```
A = np.random.randint(1,50000, size=(20000,20000))
B = np.random.randint(1,50000, size=(20000,20000))

ans = np.zeros(shape=(20000,20000))



import time
```

In [30]:

```
print(A.shape,B.shape)
```

```
(20000, 20000) (20000, 20000)
```

via loop

In [32]:

```
start = time.time()
for i in range(20000):
    for j in range(20000):
        ans[i][j] = A[i][j] + B[i][j]
calc_time = time.time() - start
print(calc_time)
```

```
502.3280107975006
```

In [33]:

```
start = time.time()
an = np.add(A,B)
calc_time = time.time() - start
print(calc_time)
```

```
1.0813589096069336
```

We can see numpy is much faster