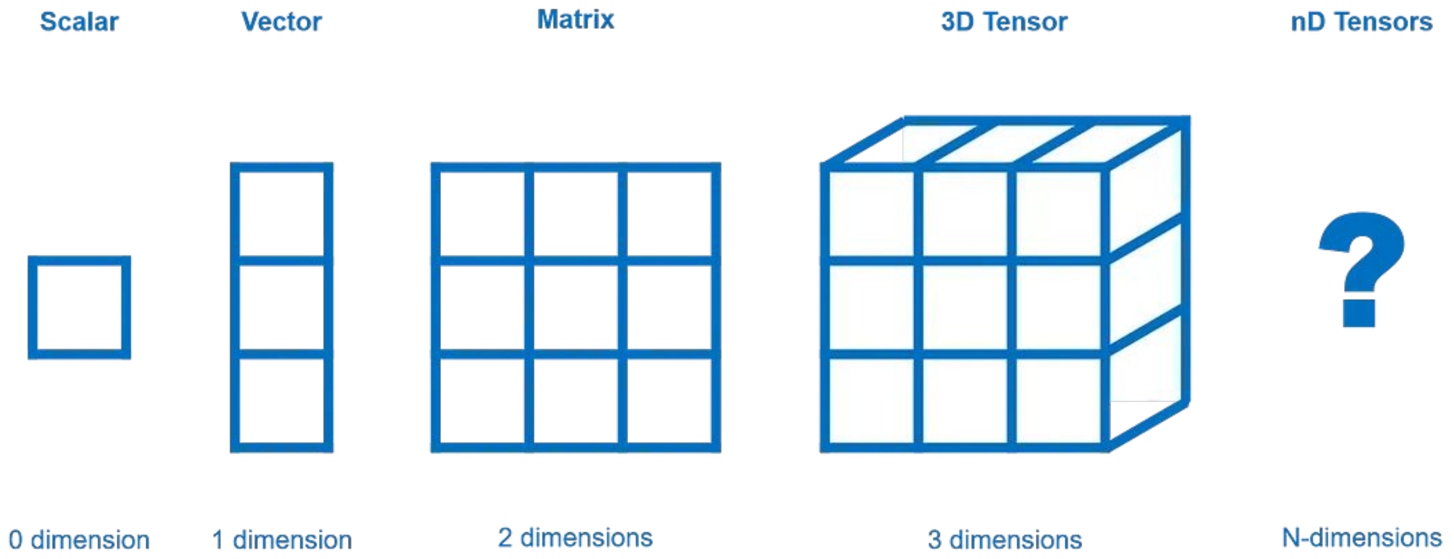
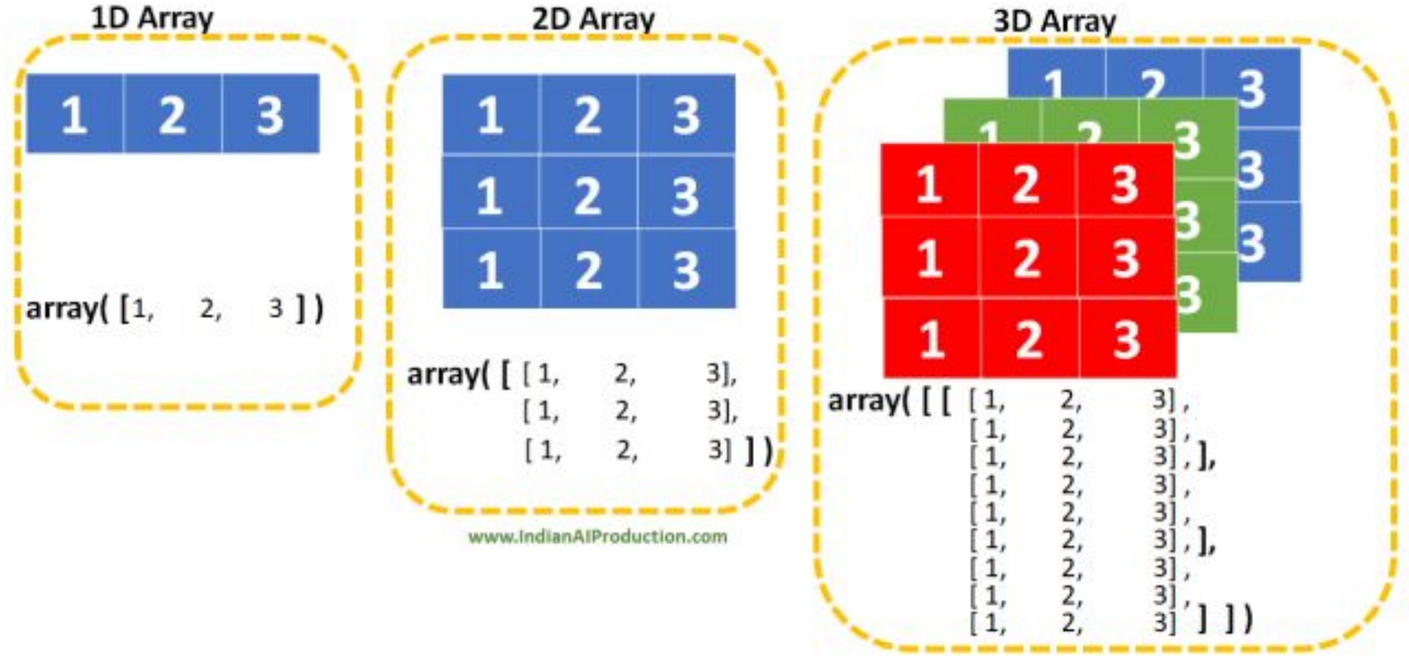
Multi Dimensional Array

Scalar, Vector , **Matrix** and **Tensor**.

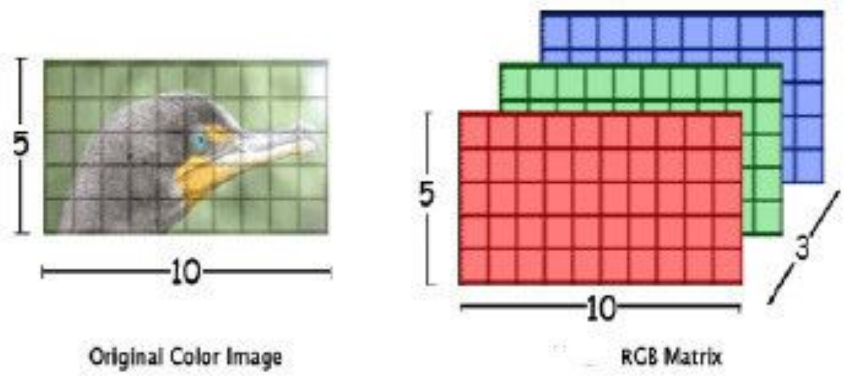


1D, 2D and 3D array



RGB image

A RGB image is actually a 3D array containing three 2D Matrices for Red, Green and Blue values.



Matrix/2D Array Examples

- Seating Arrangement in a classroom.
- Linear Algebra Usage (Matrix representation).
- Displays are 2D array of pixels.

2D MATRIX

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2 x 2   3 x 3

Dimensions of matrices
Rows x Columns

2D Matrix in Python using Numpy

```
import numpy as np
arr2D = np.zeros(shape=(row,col), dtype = datatype)
```

`arr2D = np.zeros(shape=(3,4), dtype = int)`
`print(arr2D)`
	Output
	`[[0 0 0 0]`
	` [0 0 0 0]`
	` [0 0 0 0]]`

`arr2D = np.zeros(shape=(3,4), dtype = float)`
`print(arr2D)`
	Output
	`[[0. 0. 0. 0.]`
	` [0. 0. 0. 0.]`
	` [0. 0. 0. 0.]]`

`arr2D = np.zeros(shape=(3,4), dtype = str)`
`print(arr2D)`
	Output
	`[['' '' '' '']`
	` ['' '' '' '']`
	` ['' '' '' '']]`

2D Matrix from Python List using Numpy

`arr2D=np.array( [[1,2,4,6],[5,7,9,8]] )`
`print(arr2D)`
`print(arr2D.dtype)`
	Output
	`[[1 2 4 6]`
	` [5 7 9 8]]`
	`int64`

`arr2D=np.array( [[1,2,4.2,6],[5,7,9,8]] )`
`print(arr2D)`
`print(arr2D.dtype)`
	Output
	`[[1.  2.  4.2 6. ]`
	` [5.  7.  9.  8. ]]`
	`float64`

Indexing in 2D Matrix

- **a[row_num]** will give us a single linear array.
- **a[row_num][col_num]** can access individual cell.

a	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

```
a = np.array( [[1,2,4,6],[5,6,7,8],[9,0,1,2]] )
print(a)
print("First Row", a[0])
print("Second Row", a[1])
print("Third Cell", a[0][2])
```

Output

```
[[1 2 4 6]
 [5 6 7 8]
 [9 0 1 2]]
First Row [1 2 4 6]
Second Row [5 6 7 8]
Third Cell 4
```

Accessing the shape and size of a 2D Matrix

```
arr2D = np.zeros(shape=(3,4), dtype = int)
print(arr2D)
row, col = arr2D.shape
total_cells = arr2D.size
print(“Row Amount”, row)
print(“Column Amount”, col)
print(“Total cells”, total_cells)
```

Output

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
Row Amount 3
Column Amount 4
Total cells 12
```

Iteration of 2D Matrix

```
arr=np.array( [[1,2,4,6],[5,7,9,8]] )
r_len , c_len = arr.shape
#The Outer loop iterates rows
for r in range(r_len):
#The inner loop iterates column of each row
    for c in range(c_len):
        print(arr[r][c],end=' ')
    print()
```

Output

```
1 2 4 6
5 7 9 8
```

Scalar & Matrix Multiplication

$$\alpha A = 2 \cdot \begin{bmatrix} 0 & 2 & 3 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2\cdot 0 & 2\cdot 2 & 2\cdot 3 \\ 2\cdot 1 & 2\cdot 1 & 2\cdot 0 \end{bmatrix} = \begin{bmatrix} 0 & 4 & 6 \\ 2 & 2 & 0 \end{bmatrix}$$

```
A=np.array( [[0,2,3],[1,1,0]] )
print("Before",A)
r_len , c_len = A.shape
#The outer loop iterates rows
for r in range(r_len):
#The inner loop iterates column of each row
    for c in range(c_len):
        A[r][c] = 2*A[r][c]
print("After",A)
```

Output
Before:
[[0 2 3]
 [1 1 0]]
After:
[[0 4 6]
 [2 2 0]]

Matrix Multiplication Visual

m = 3
p = 4
n = 3

```
for i in range(m):
    for j in range(p):
        sum=0
        for k in range(n):
            sum += array_1[i][k]*array_2[k][j]
        result[i][j] = sum
```

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \\ 2 & 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 5 & 1 & 1 \\ 6 & 7 & 4 & 2 \\ 1 & 8 & 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} \quad = 1x2 + 2x6 + 1x1 \quad \\ \\ \\ \end{bmatrix}$$

Matrix Multiplication Visual

m = 3
p = 4

```
for i in range(m):
    for j in range(p):
        sum=0
        for k in range(n):
            sum += array_1[i][k]*array_2[k][j]
        result[i][j] = sum
```

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \\ 2 & 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 5 & 1 & 1 \\ 6 & 7 & 4 & 2 \\ 1 & 8 & 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} \quad & \quad & \quad & \quad \\ & & & \\ & & & \end{bmatrix}$$

Matrix Multiplication Visual

m = 3
p = 4

```
for i in range(m):
    for j in range(p):
        sum=0
        for k in range(n):
            sum += array_1[i][k]*array_2[k][j]
        result[i][j] = sum
```

$$\begin{bmatrix} \boxed{1 \quad 2 \quad 1} \\ 0 \quad 1 \quad 0 \\ 2 \quad 3 \quad 4 \end{bmatrix} \times \begin{bmatrix} 2 & 5 & 1 & 1 \\ 6 & 7 & 4 & 2 \\ 1 & 8 & 3 & 4 \end{bmatrix}$$

=

Matrix Multiplication Visual

m = 3
p = 4

```
for i in range(m):
    for j in range(p):
        sum=0
        for k in range(n):
            sum += array_1[i][k]*array_2[k][j]
        result[i][j] = sum
```

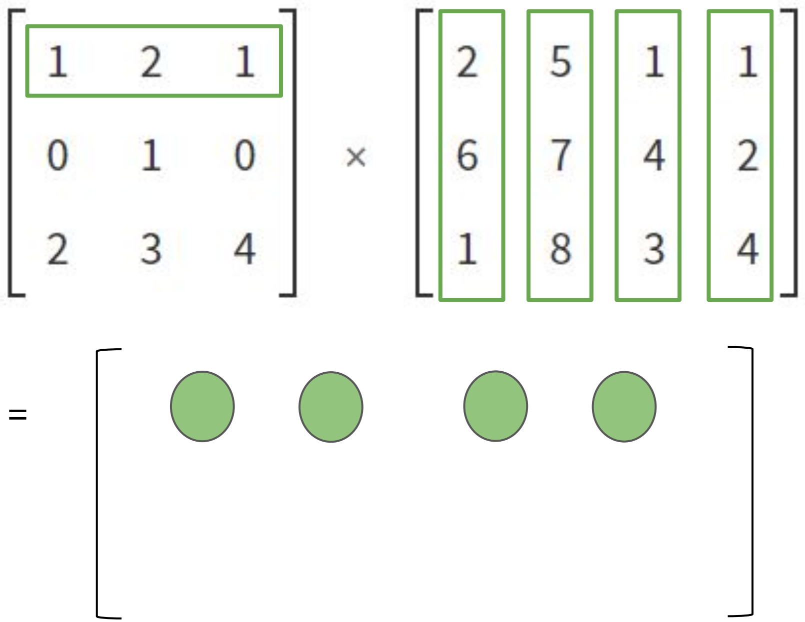
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \\ 2 & 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 5 & 1 & 1 \\ 6 & 7 & 4 & 2 \\ 1 & 8 & 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} \quad & \quad & \quad & \quad \\ & & & \\ & & & \end{bmatrix}$$

Matrix Multiplication Visual

m = 3
p = 4

```
for i in range(m):
    for j in range(p):
        sum=0
        for k in range(n):
            sum += array_1[i][k]*array_2[k][j]
        result[i][j] = sum
```



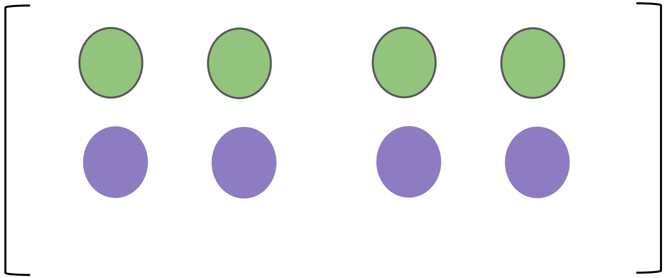
Matrix Multiplication Visual

m = 3
p = 4

```
for i in range(m):
    for j in range(p):
        sum=0
        for k in range(n):
            sum += array_1[i][k]*array_2[k][j]
        result[i][j] = sum
```

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \\ 2 & 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 5 & 1 & 1 \\ 6 & 7 & 4 & 2 \\ 1 & 8 & 3 & 4 \end{bmatrix}$$

=



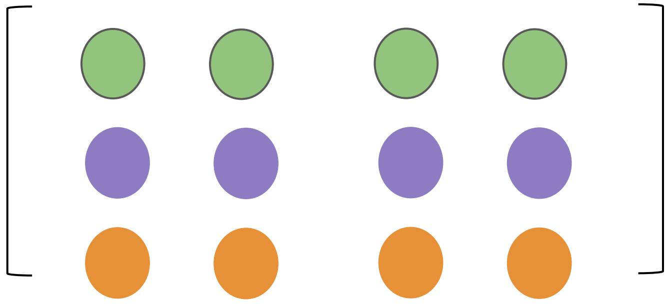
Matrix Multiplication Visual

m = 3
p = 4

```
for i in range(m):
  for j in range(p):
    sum=0
    for k in range(n):
      sum += array_1[i][k]*array_2[k][j]
    result[i][j] = sum
```

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 0 \\ 2 & 3 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 5 & 1 & 1 \\ 6 & 7 & 4 & 2 \\ 1 & 8 & 3 & 4 \end{bmatrix}$$

=



Matrix & Matrix Multiplication Code


```
A=np.array( [[0,5],[1,2],[3,4]] )
B=np.array( [[3,2,1],[1,2,4]] )
print("A(3,2):")
print(A)
print("B(2,3):")
print(B)
Ar_len , Ac_len = A.shape[0], A.shape[1]
Br_len , Bc_len = B.shape[0], B.shape[1]
#Here, Bc_len==1 and Ac_len==Br_len
C=np.zeros( shape=(Ar_len,Bc_len), dtype=int)
Cr_len, Cc_len = C.shape[0], C.shape[1]
for i in range(Cr_len):
    for j in range(Cc_len):
        sum=0
        for k in range(Br_len):
           sum += A[i][k] * B[k][j]
        C[i][j] = sum
print("A X B = C(3,3):")
print(C)
```

OUTPUT

```
A(3,2):
[[0 5]
 [1 2]
 [3 4]]

B(2,3):
[[3 2 1]
 [1 2 4]]

A X B = C(3,3):
[[ 5 10 20]
 [ 5  6  9]
 [13 14 19]]
```