21-02-2026

Agenda!

→ Numpy-3 ●

→ pandas

Numpy → indexing ●

Element wise

```
a = np.array([1,2,3])
b = np.array([4,5,6])
a+b
```

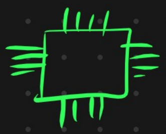
→ same operation in python uses loops but in numpy it is vectorization.

+, -, * → numpy → element wise operation

```
a = [1,2]
b = [3, 4]
```

```
sum = [   ]
for i,v in enemerate(a):
    s = a[i] + b[i]      → not optimised
    sum.append(s)
print(sum)
  → [4,6]
```

instruction (1)

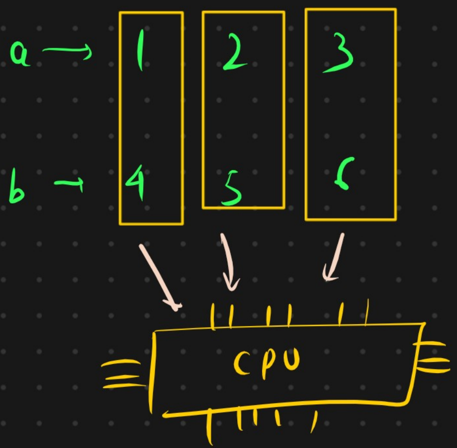
↳  → output

(1000/s)

a → 1000 → loops inefficient

b → 1000

numpy → C ← modules ← CPU



instruction ← add

parallel processing

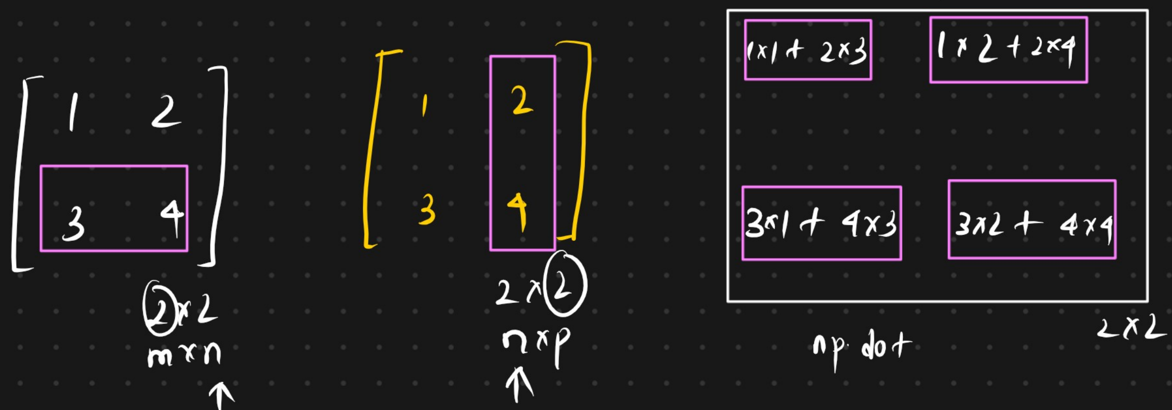
np.dot → dot product

[1, 2, 3]   [4, 5, 6]

a_1, a_2, a_3 b_1, b_2, b_3

$$\text{sum} \leftarrow \sum_{i=1}^{3} a_i \cdot b_i \rightarrow a_1 \cdot b_1 + a_2 b_2 + a_3 b_3$$

$$\rightarrow 4 + 10 + 18 \rightarrow 32$$



Facts → dot product

F-1

$A \rightarrow m \times n$

$B \rightarrow n \times p$

$A \cdot B \rightarrow m \times p$

F-2

columns of first matrix should match with rows of second matrix

Q. 4×5 , 5×8

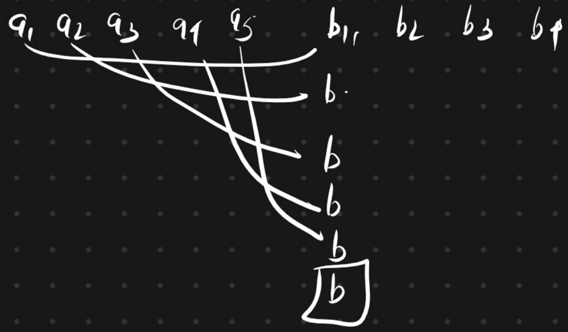
↓

4×8

Q. 4×5 , 6×4

↓

not work



1, 2, 3   4, 5, 6

↓   ↓

1×3 1×3

↓

Transpose (internally)

1, 2, 3   4
5
6

↓   ↓

1×3 3×1

$m \times n$ $n \times p$

$m \times p \rightarrow 1 \times 1 \rightarrow$

Q. $a = 2 \times 3$

$b = 2 \times 3$

$a \cdot b$

```
shop_sold_amazon = np.array([
    [100, 200, 20, 30, 5], # state-1-product-sold
    [200, 400, 40, 60, 10] # state-2-product-sold
])
```

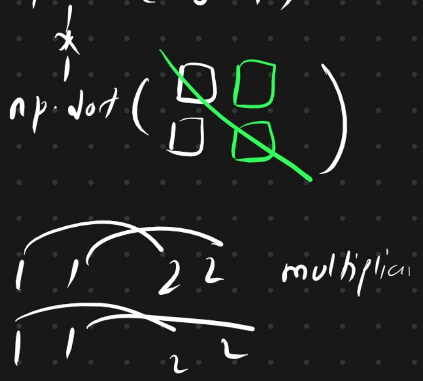
→ 2×5

Σ → total sales of state-1

```
price_amazon = np.array([
    [30, 10, 1000, 10, 200], # state-1-price
    [34, 12, 1100, 11, 220] # state-2-price
])
```

→ 2×5

$$np.eye\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$



$$np.sum\begin{bmatrix}\square & 0\\ 0 & \square\end{bmatrix} \rightarrow \text{final number}$$

$$np.sum \rightarrow \sum_{ij} X_{ij}$$

i → row
j → column
} all elements

1	1	1	1
2	1+2	1+2	3
3	1+2+3	1+2+3	6
4	1+2+3+4	1+2+3+4	10

Frecnly - (Month)

1	2	3	4	5	6	7	8	9	10	11	12
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
200	300	400	10	100	50	500	-	-	-	-	
		↓									

0 1 2 3 4 5 6 7 8 9

[3, 6, 1, 1, 1, 1, 8, 2, 5, 6] → [3 2 5 4 7 0 8 1 9 6]

1 → 2, 3, 4, 5

2 → 7

3 → 0

5 → 8

6 → 1, 9

8 → 6

1, 1, 1, 1, 2, 3, 5, 6, 6, 8

0 1

1, 1

argsort

bincount → ints

grapes

apple

banana

apple

kiwi

kiwi

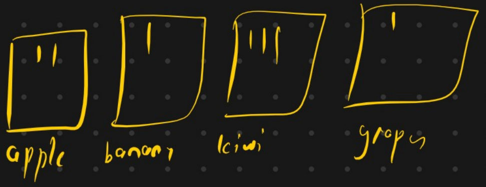
kiwi

apples → 2

banana → 1

kiwi → 3

grapes → 1

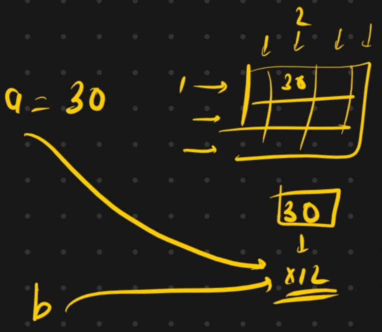


1, 2, 3, 3, 4, 5, 8 → max → 8

→ range(0, max+1)

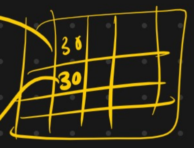
0 1 2 3 4 5 ... 8

b = a



b = a

b = x12



a = 30

b = a.copy()

numpy → def

→ OOP

→ libraries

→ except

→ creat ,

programming language