$$x = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 3 & 4 & 5 \end{bmatrix}$$

$$= \begin{bmatrix} [1 & 2 & 3] + [3 & 4 & 5] \\ [4 & 5 & 6] + [3 & 4 & 5] \\ [7 & 8 & 9] + [3 & 4 & 5] \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 6 & 8 \\ 7 & 9 & 11 \\ 10 & 12 & 14 \end{bmatrix}$$

$v_1 = [1, 2, 3, 4]$

$v_2 = [-2, 3, -1, 5]$

$v_1 \cdot v_2$

$= 1 \times (-2) + 2 \times 3 + 3 \times (-1) + 4 \times 5$

$= -2 + 6 - 3 + 20$

$= 21$

$\vec{v} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}_{1\times 3}$ $\quad M = \begin{bmatrix} 10 & 9 \\ 12 & 8 \\ 13 & 7 \end{bmatrix}_{3\times 2}$ $\quad = \begin{bmatrix} \vec{v}\cdot\vec{m}_1 & \vec{v}\cdot\vec{m}_2 \end{bmatrix}_{1\times 2}$

$= \begin{bmatrix} 73 & 46 \end{bmatrix}$

$1\times 10 + 2\times 12 + 3\times 13$

$= 10 + 24 + 39$

$= 73$

$1\times 9 + 2\times 8 + 3\times 7$

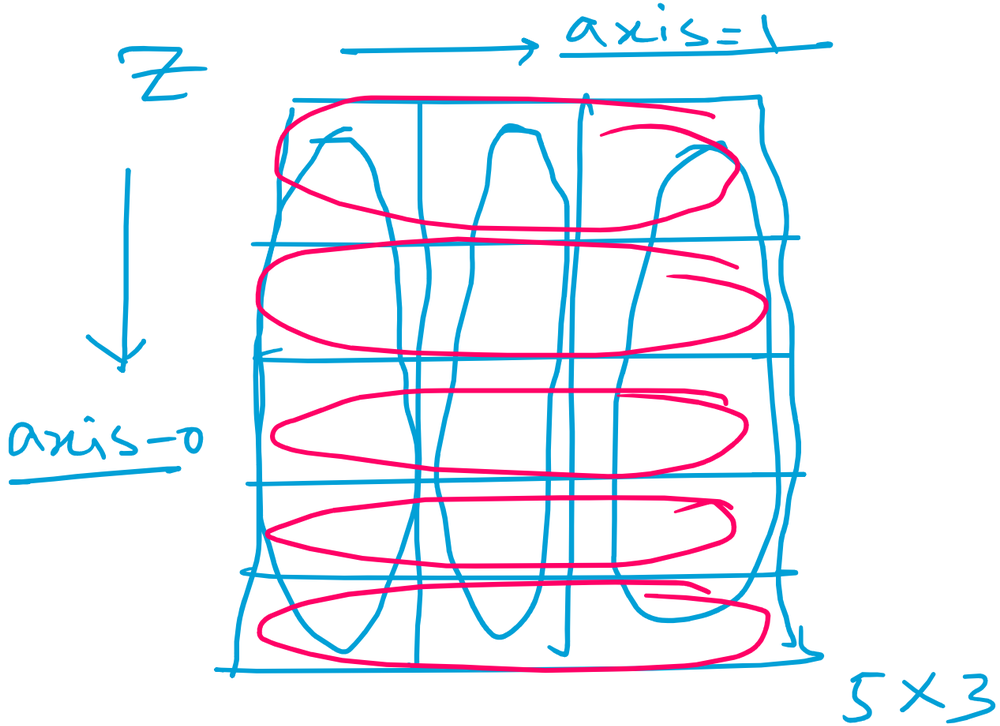
$= 9 + 16 + 21$

$= 46$

$M_1 \times M_2$

$M_1 = \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \end{bmatrix}_{2\times 4}$ $\quad \times \quad M_2 = \begin{bmatrix} \vec{a} & \vec{b} & \vec{c} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}_{4\times 3}$

$= \begin{bmatrix} \vec{v}_1\cdot\vec{a} & \vec{v}_1\cdot\vec{b} & \vec{v}_1\cdot\vec{c} \\ \vec{v}_2\cdot\vec{a} & \vec{v}_2\cdot\vec{b} & \vec{v}_2\cdot\vec{c} \end{bmatrix}_{2\times 3}$



Z.sort(axis=0) → columnwise sorting

Z.sort(axis=1) → rowwise sorting.

axis argument inside the numpy methods actually mean along which axis it will consider the vectors to be.

axis-0: The vectors are aligned columnwise (all rows)

axis-1: The vectors are aligned rowwise (all columns)

argsort (arg - argument)

	0	1	2	3	4	5
x =	3	-1	4	8	0	2

Sort

[-1, 0, 2, 3, 4, 8]

argsort

[1, 4, 5, 0, 2, 3]

argsort will return array containing indices of numbers in the original array such that the first number will be the index of lowest number.

second " " " " " second lowest "

third " " " " " third " "

. . . . . .

Cumulative sum

$x = [2, 3, 0, 1, 4]$

$x.cumsum() / y = [2, 5, 5, 6, 10]$

$x = [x_0, x_1, x_2, \ldots, x_n]$

$x.cumsum()$ or $np.cumsum(x) = y$

$y = [y_0, y_1, y_2, \ldots, y_n]$

$$y_k = y_{k-1} + x_k \quad | \quad y_0 = x_0$$

$y_1 = y_0 + x_1$ $\quad$ $y_3 = y_2 + x_3$

$y_2 = y_1 + x_2$ $\quad$ $y_4 = y_3 + x_4$

Cumulative product

$x = [2, 3, 1, 4, 5]$

$z = cumprod(x)$

$z = [2, 6, 6, 24, 120]$

$$z_k = z_{k-1} \times x_k \quad | \quad z_0 = x_0$$

0 1 2 3 4

$x = [2, 3, 0, 1, 4]$

0 1 2 3 4

$y = [2, 5, 5, 6, 10]$

Variance & Std deviation

$x = [2, 3, 0, 1, 9]$

$$\text{mean} = \frac{2+3+0+1+9}{5}$$

$$= \frac{5+0+10}{5} = \frac{15}{5} = 3$$

$x - \text{mean}$

$= [2-3,\ 3-3,\ 0-3,\ 1-3,\ 9-3]$

$= [-1,\ 0,\ -3,\ -2,\ 6]$

$(x - \text{mean})^2 = [1, 0, 9, 4, 36]$

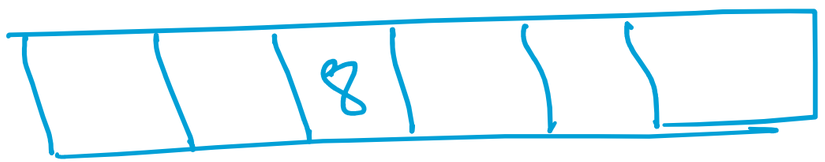
$$\frac{\text{sum}(x-\text{mean})^2}{n} = \frac{1+9+4+36}{5} = \frac{50}{5} = 10$$

variance.

Std dev = $\sqrt{\text{variance}}$

$= \sqrt{10}$

x =
y =

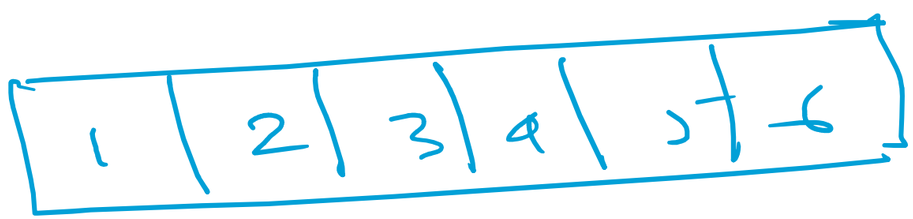


x points to the memory location where the numbers are stored.

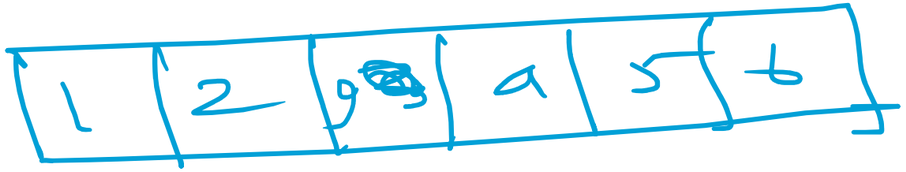
y = x → y will point to the same memory location

y = x.copy( )

x =



y =



Stacking :-

$X =$

x_1	x_2	x_3	x_4
x_5	x_6	x_7	x_8

2×4

$Y =$

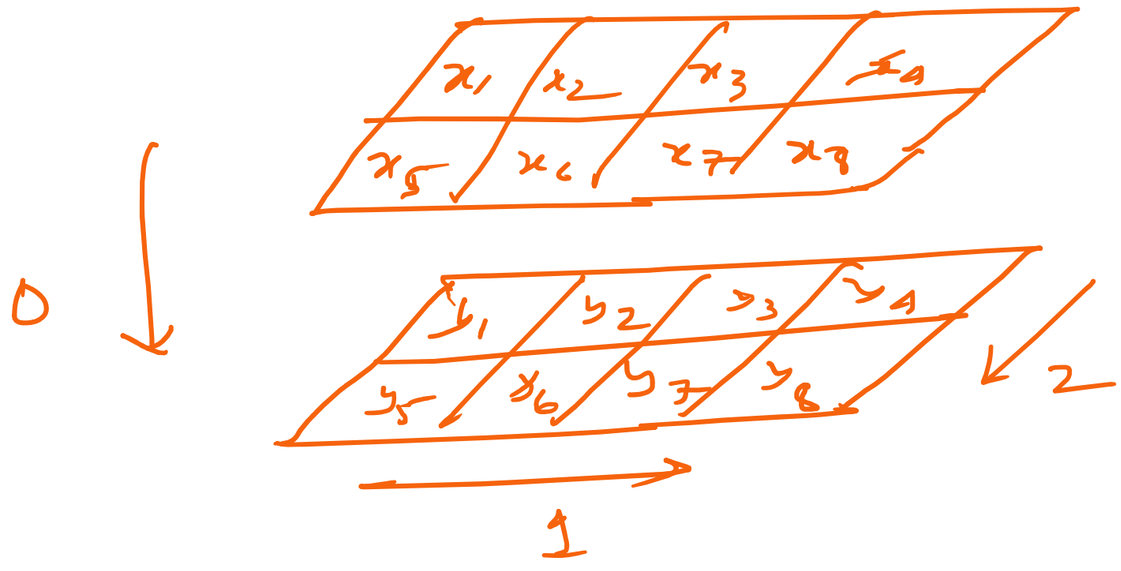
y_1	y_2	y_3	y_4
y_5	y_6	y_7	y_8

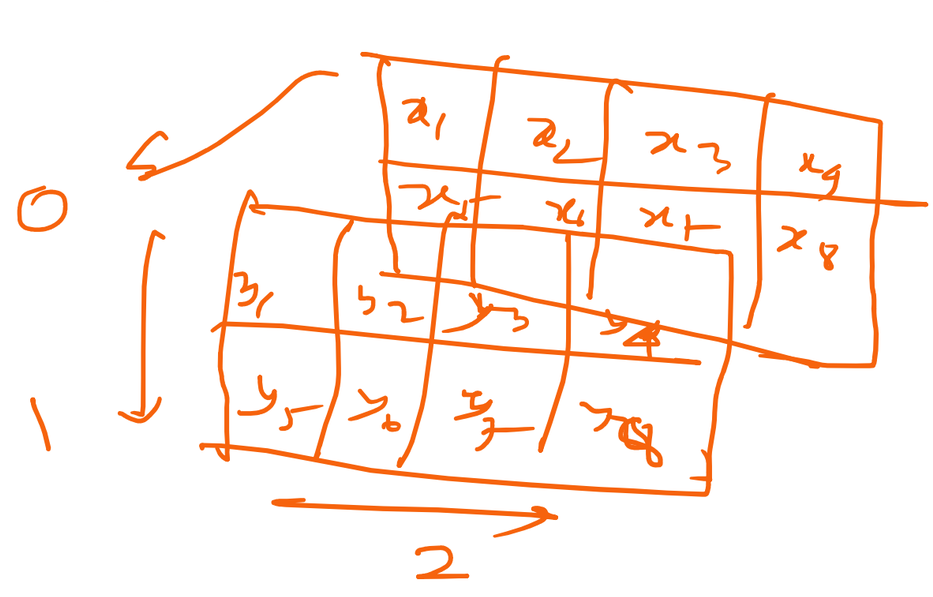
2×4

`Z = np.stack((X,Y), axis=0)`

`Z.shape = (2,2,4)`

`W = np.stack((X,Y), axis=1)`





Concatenation:

$X =$

x_1	x_2	x_3	x_4	x_5
x_6	x_7	x_8	x_9	x_{10}

2×5

$Y =$

y_1	y_2	y_3	y_4	y_5
y_6	y_7	y_8	y_9	y_{10}

2×5

3×5

Z = np.concatenate((X, Y), axis=0)

$Z =$

x_1	x_2	x_3	x_4	x_5
x_6	x_7	x_8	x_9	x_{10}
y_1	y_2	y_3	y_4	y_5
y_6	y_7	y_8	y_9	y_{10}

4×5

(col of X = col. of Y)

W = np.concatenate((X, Y), axis=1)

x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3	y_4	y_5
x_6	x_7	x_8	x_9	x_{10}	y_6	y_7	y_8	y_9	y_{10}

(row no. of X = row no. of Y)