TEORIA

VETORES

Vetor: An ordered list of numbers

Dimensionality: the number of numbers (elements)

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \neq \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

"Column Vectors"

$[\,1\ 3\ 2\ 5\ 4\,]$ → "Row Vectors"

$[\,-1\ -14\,]$ ↗

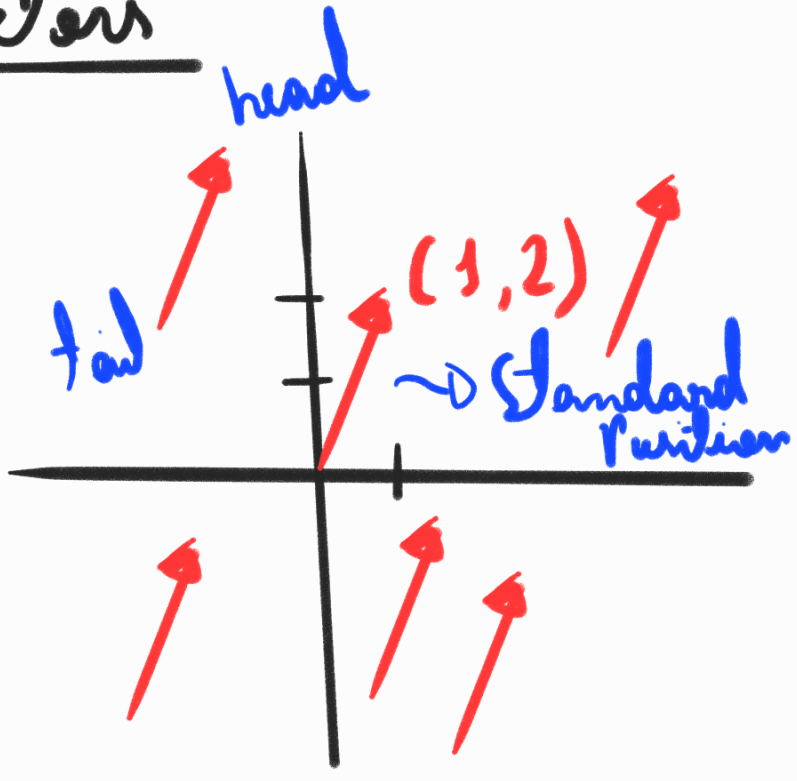
Vector Notation

$$\begin{bmatrix} 1 \\ 0 \\ 5 \\ -2 \end{bmatrix} = V \text{ or } \vec{v} \text{ or } \vec{\mathbf{v}} \text{ or } \mathbf{V}$$

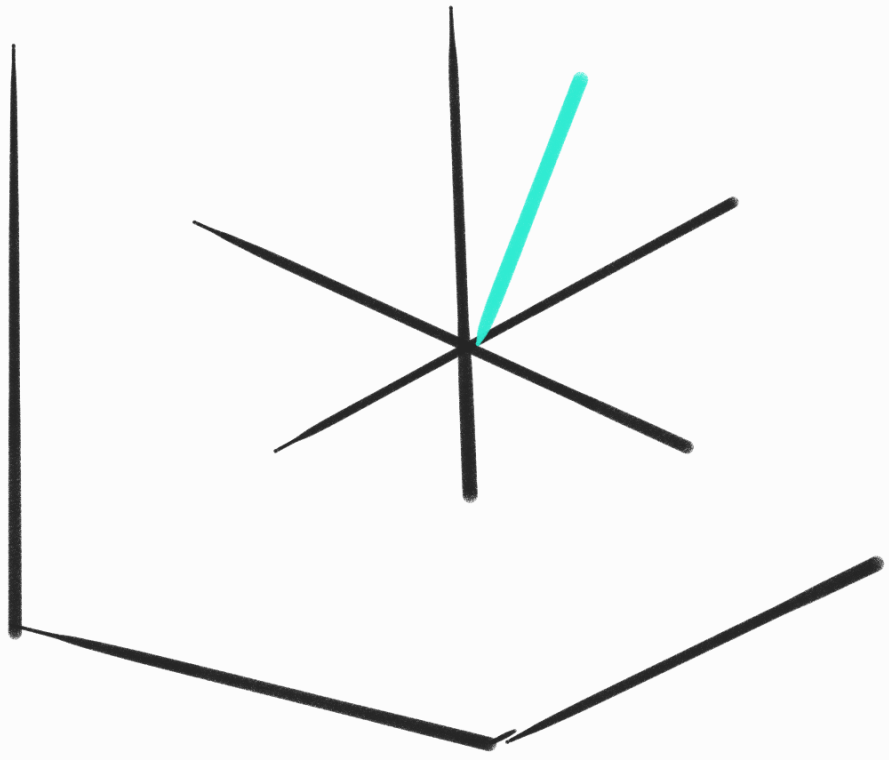
↳ Italic letter

Geometric Vectors

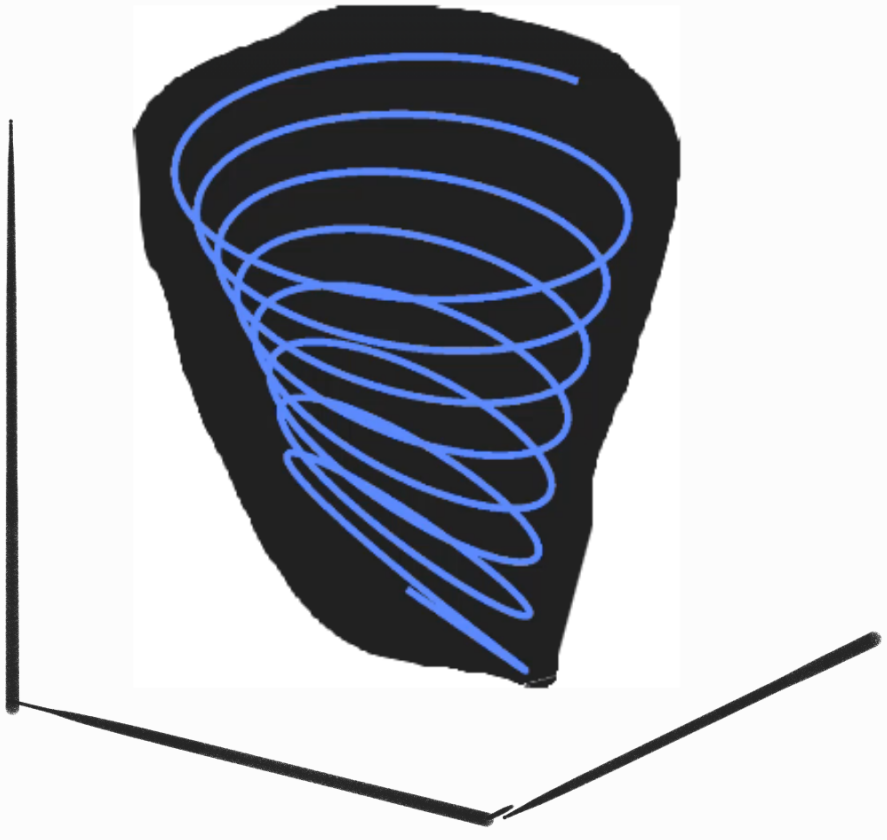
$$[1 \quad 2]$$



$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$



$$\begin{bmatrix} \sin(x) \\ x\cos(x) \\ x \end{bmatrix}$$



Creating Vectors in Python

```
V2 = [3, -2]        (2D) x = 3 & y = -2
V3 = [4, -3, 2]     (3D) x = 4, y = -3 & z = 2
```

transpose (Row to Column (or vice-versa))

```
V3T = np.transpose(V3)
```

Creating Graphics

2D

```
plt.plot([0, V2[0]], [0, V2[1]])
```

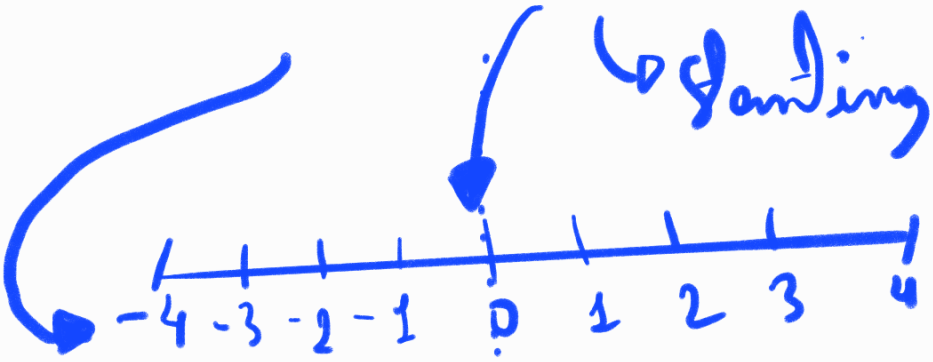
it starts at the origin

```
plt.axis('equal')
```

Ensures that the scale of the X-axis is equal to that of the Y-axis.

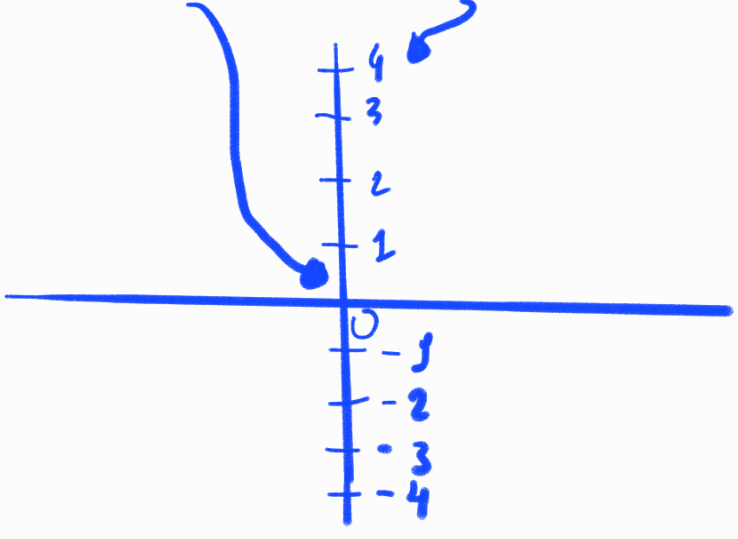
plt.plot([-4,4], [0,0], 'k--') → Previous line

↳ Starting



this draws a horizontal line starting -4 & 4 on the X-axis, positioned at 0 on the Y-axis

plt.plot([0,0], [-4,4], 'k--') → Previous line



plt.grid() → Add lines.

to facilitate the reading of coordinates

→ this command turns on the background grid lines

```
plt.axis((-4,4,-4,4))
```

this sets the boundaries of your graph's view. It tells Python exactly where to "cut" the camera frame.

(x_min, x_max, y_min, y_max)

```
fig = plt.figure(figsize = plt.figaspect(1))
```

this encodes a new drawing area (a "figure")

this is a clever trick to make the plot window a perfect square. In linear Algebra, this is important so that the perpendicular of your 3D vectors aren't distorted

```
ax = plt.axes(projection = '3d')
```

this tells Matplotlib that you aren't drawing on a flat paper anymore, but in a 3D space with X, Y and Z axes. It create the depth need to see "inside" the coordinate system.

ax.plot([0, v3[0]], [0, v3[1]], [0, v3[2]], linewidth=3)

↳ this defines the start and end points for the three coordinates (x, y, z).

→ It start at the origin (0, 0, 0)

→ It ends at your vector's values (4, -3, 2)

→ this makes the vector line thicker and easier to see in the 3D view.

```
ax.plot([0,0],[0,0],[-4,4],'k--')
ax.plot([0,0],[-4,4],[0,0],'k--')
ax.plot([-4,4],[0,0],[0,0],'k--')
```

x y z

plt.show()

this renders all the elements (the 3D axes and your vector) into the final interactive window.