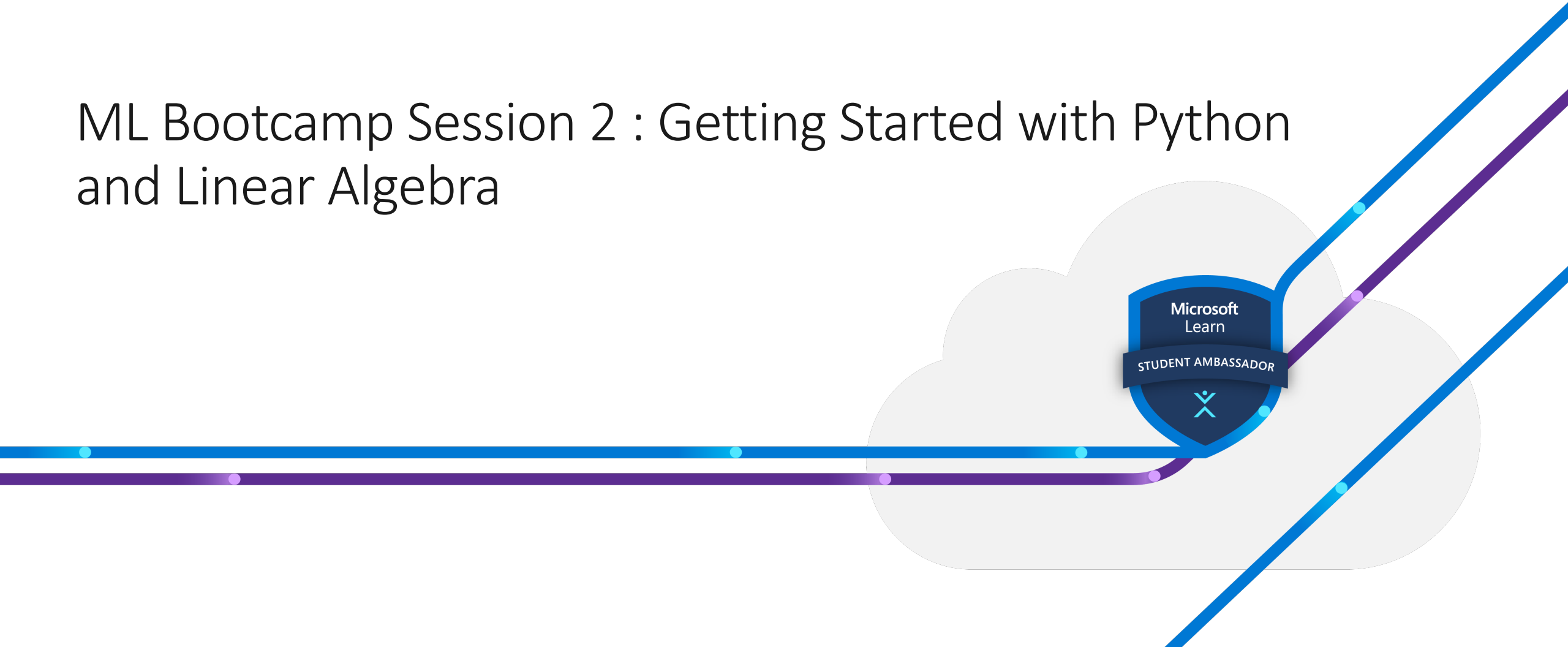
Microsoft Learn
Student Ambassadors

ML Bootcamp Session 2 : Getting Started with Python and Linear Algebra



Python Setup (Local)

To setup python on your local device:

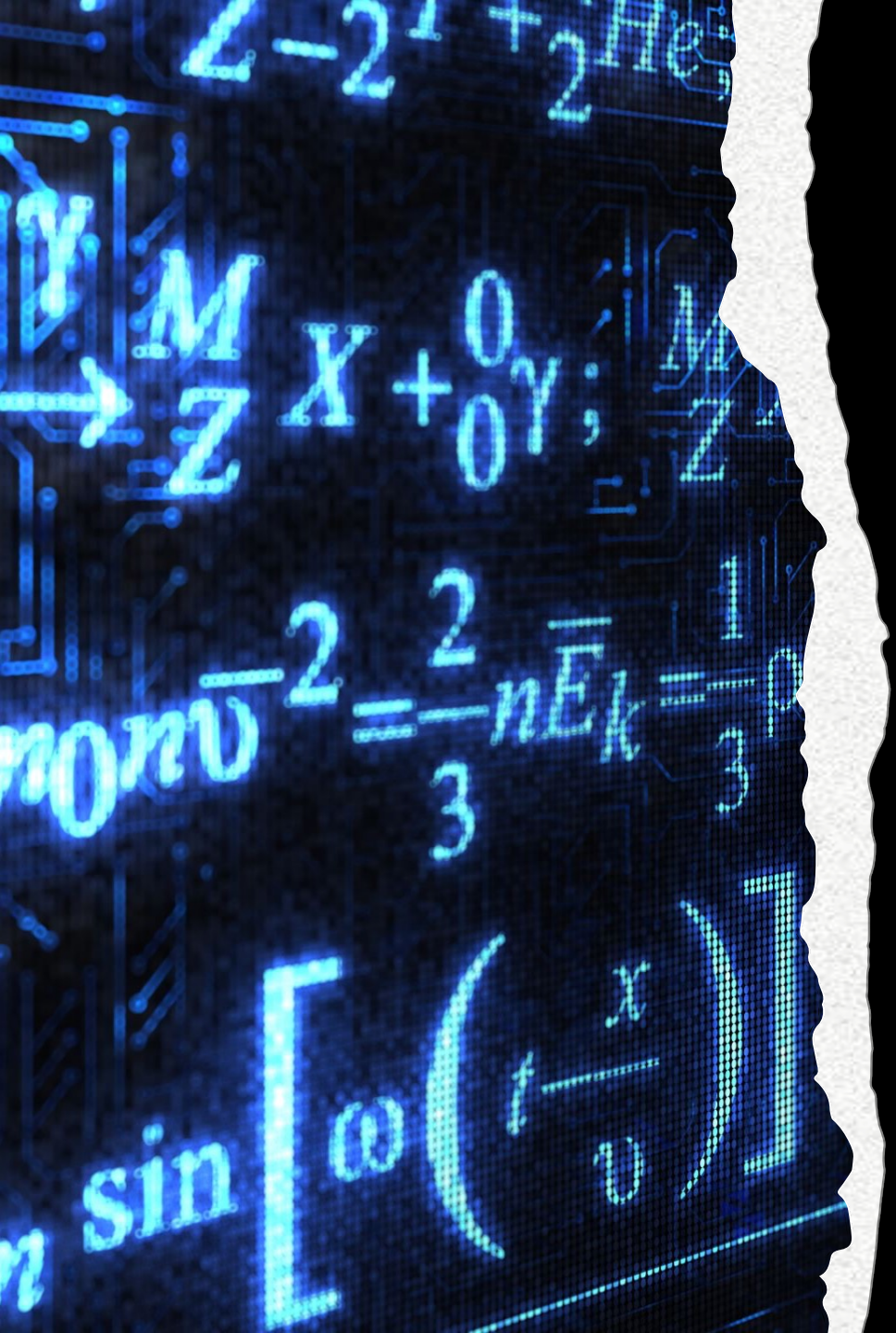
1. Download python from python.org
2. Use the packaged Python Idle as Integrated development environment.

OR

1. Download Anaconda Installer / Miniconda
2. Choose your IDE:
   i. VS CODE
   ii. Jupyter Notebook(Anaconda)
   iii. Spyder
   iv. JupyterLab

Python Setup (Online)

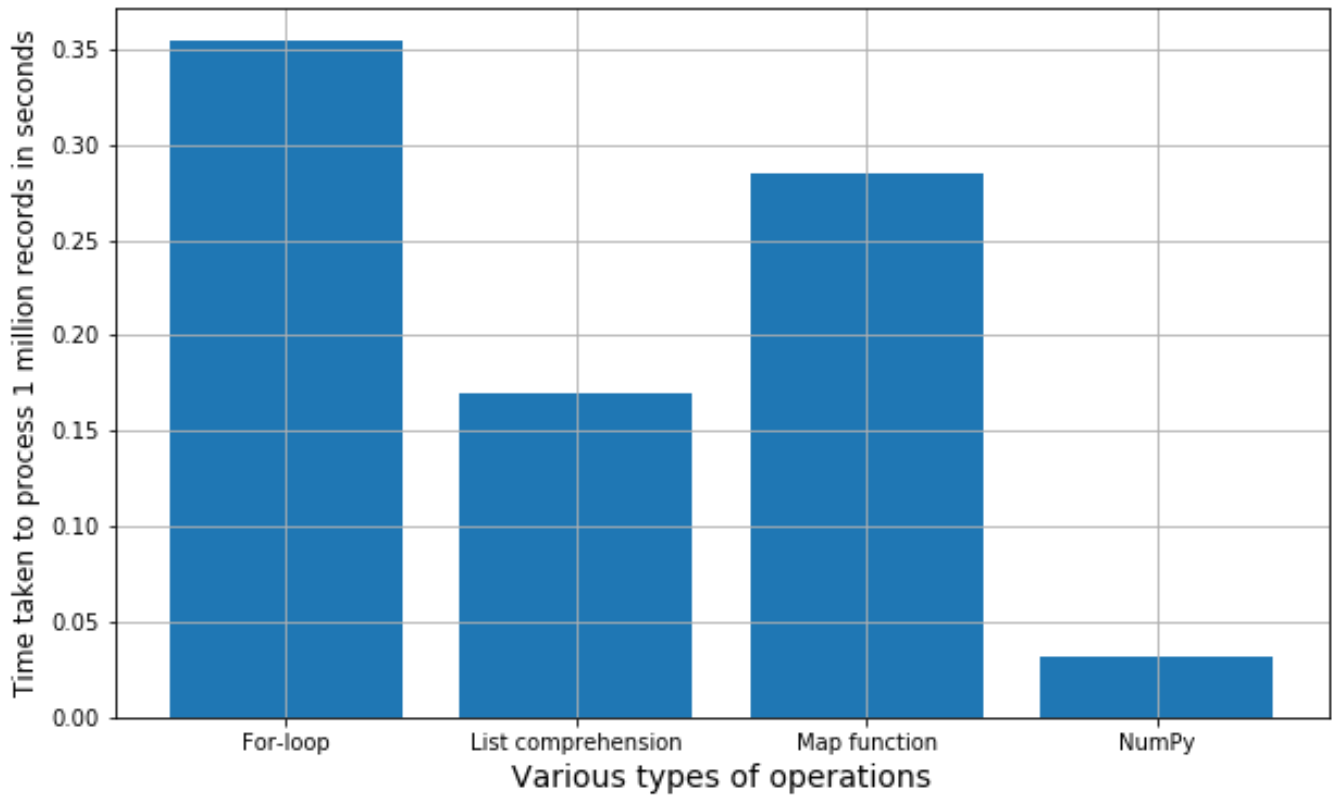
Using colab.research.google.com



LINEAR ALGEBRA

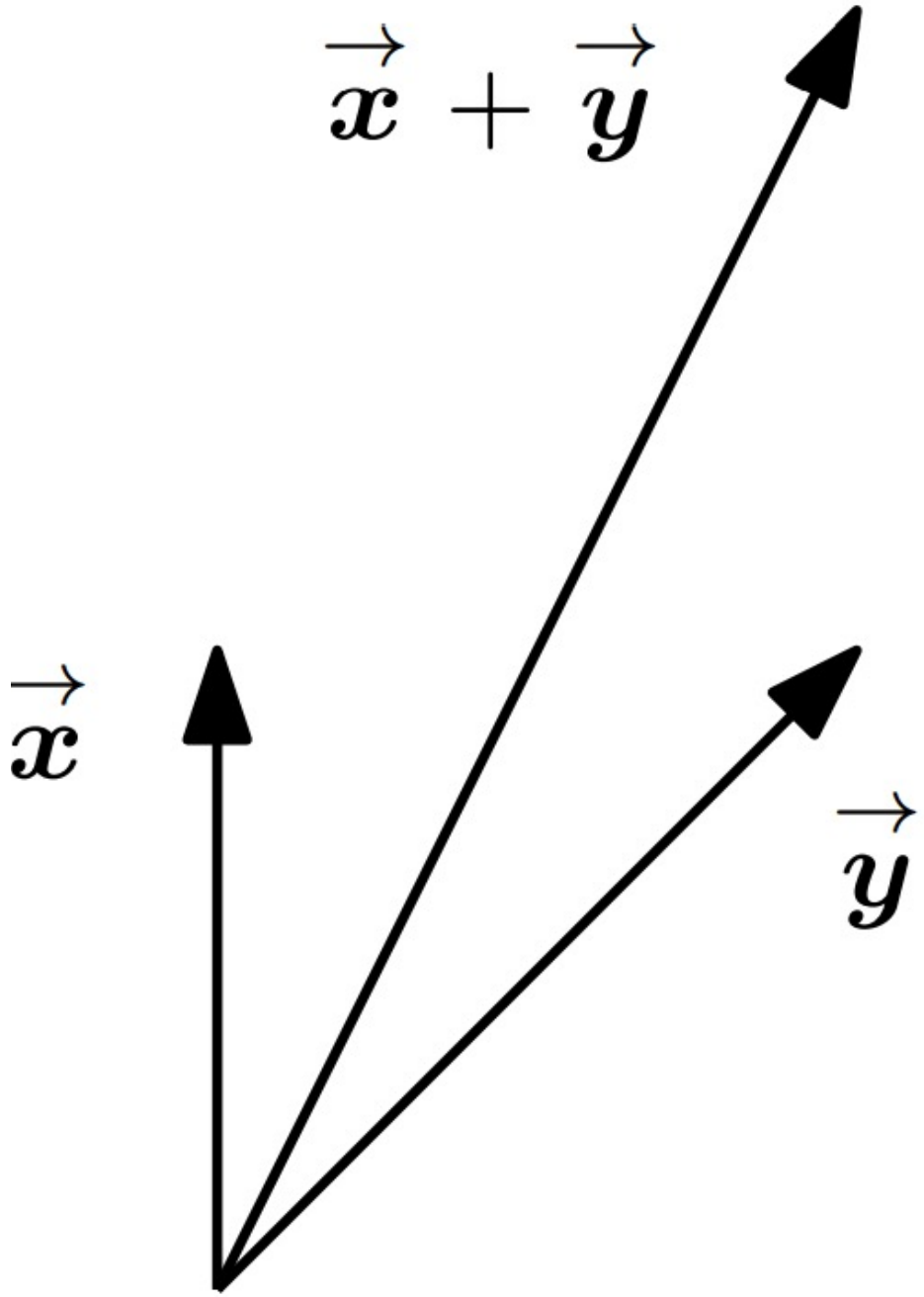
Why Vectors?

SPEED



Why Vectors?

- SPEED
- Organized Data
- Mathematically Efficient
- Supports GPU
- (And is building block for DL)

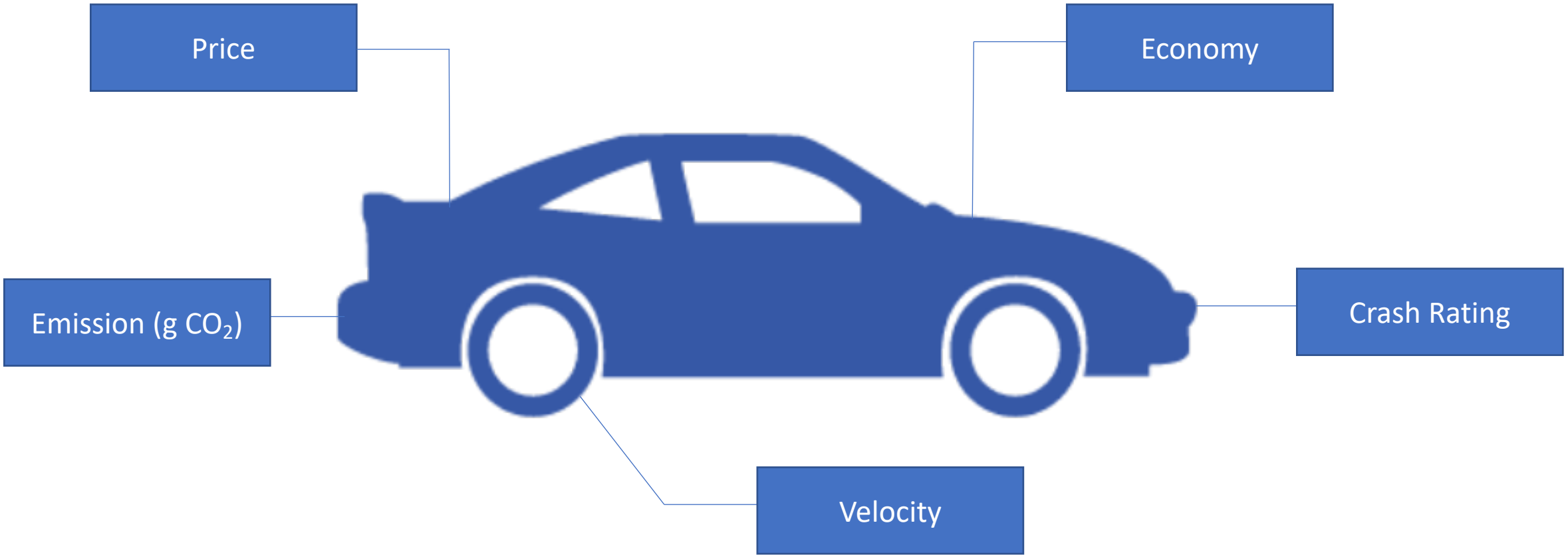


Vector

- can be added, such that
- $\vec{x} + \vec{y} = \vec{z}$

multiplication by a scalar
$\lambda\vec{x}$

Vector (Intuition)



Vector (Intuition)

5 Dimension

$$\begin{bmatrix} 13 \\ 21 \\ 4 \\ 210 \\ 121 \end{bmatrix}$$

- 13 → ₹ (in Lakh)
- 21 → (in Km/L)
- 4 → (out of 5)
- 210 → (Top Speed Km/hr)
- 121 → (g CO_2)

```
[1]  # Import NumPy: Numerical Python Library
     import numpy as np

     # Make a vector from list
     v = np.array([1, 2, 3, 4])
     print(v)
```

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

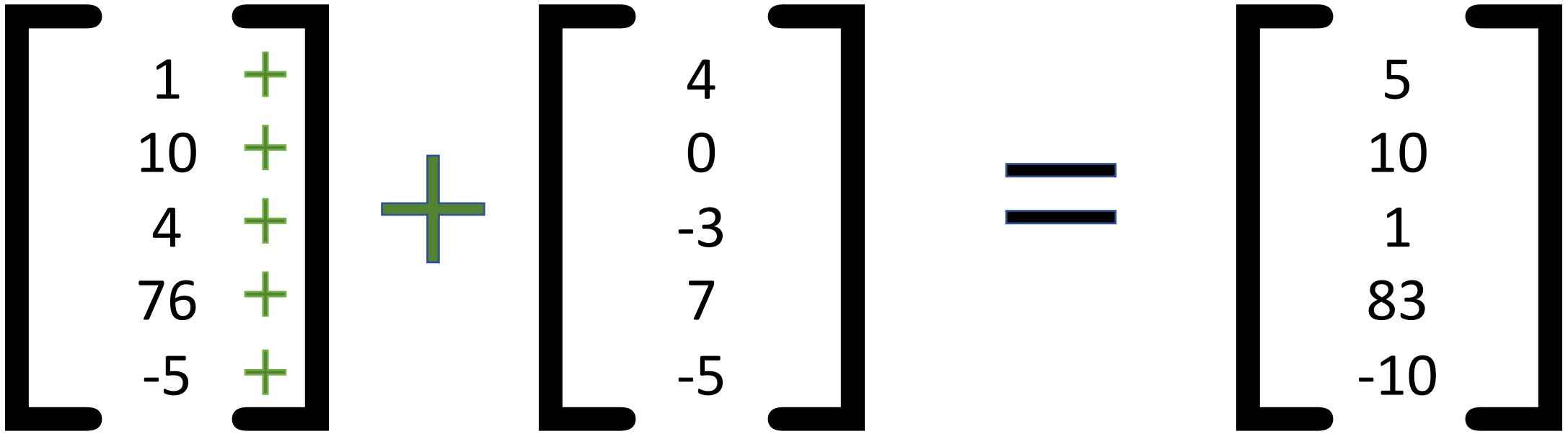
```
[2]  # Make same from range function
     v = np.array(range(1, 5))
     print(v)
```

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

Vector (Definition)

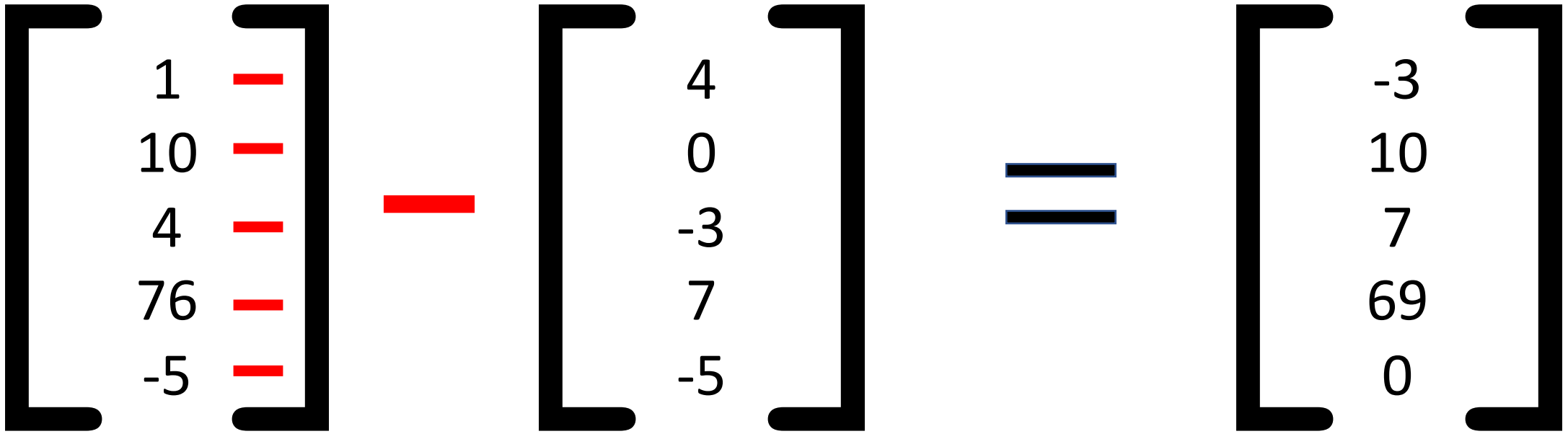
Vector Arithmetic

Addition



Vector Arithmetic

Subtraction



Vector Arithmetic

Subtraction

$$\begin{bmatrix} 1 - 4 \\ 10 - 0 \\ 4 - (-3) \\ 76 - 7 \\ -5 - (-5) \end{bmatrix} =$$

```
[5]  # vector subtraction
     from numpy import array
     # define first vector
     a = array([1, 10, 4, 76, -5])
     print(a)
     # define second vector
     b = array([4, 0, -3, 7, -5])
     print(b)
     # subtract vectors
     c = a - b
     print(c)
```

```
[ 1 10  4 76 -5]
[ 4  0 -3  7 -5]
[-3 10  7 69  0]
```

Vector Arithmetic

Multiplication by a scalar

```
[6]  # Multiplication by a scalar
     import numpy as np
     # define a vector
     v = np.array([1, 10, 4, 76, -5])
     # multiply with scalar
     print(2*v)
```

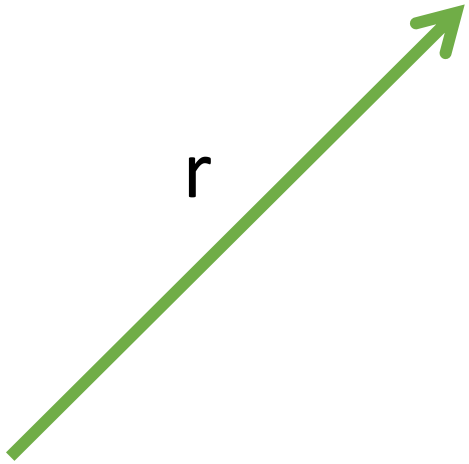
```
     [  2  20   8 152 -10]
```

```
[7]  # multiplying with a vector of same length as v
     # and having same value of all elements
     scalar_vec = np.array([2, 2, 2, 2, 2])

     print(scalar_vec*v)
```

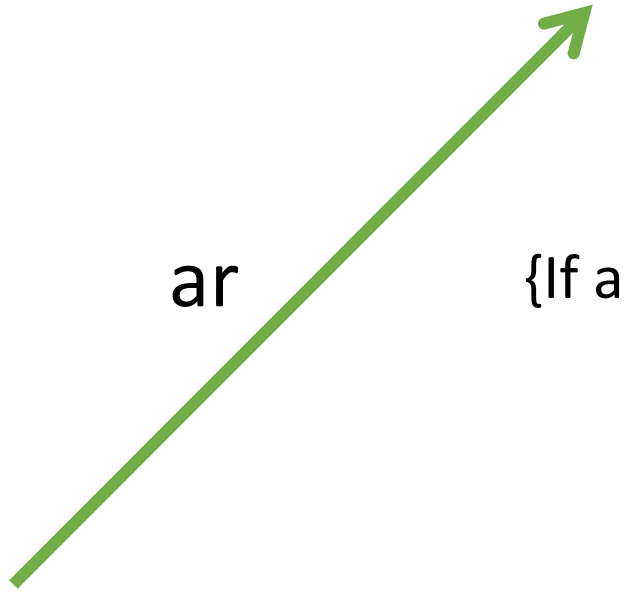
```
     [  2  20   8 152 -10]
```

Vector Arithmetic

Multiplication by a scalar



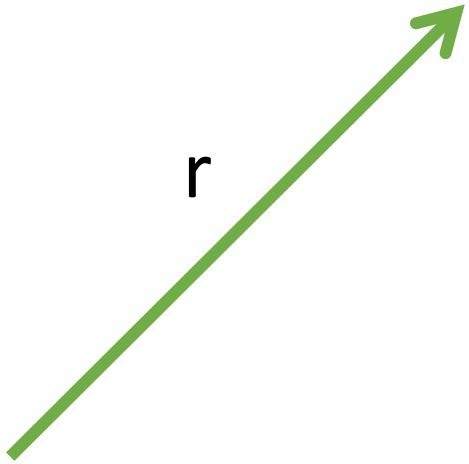
Multiply by a;



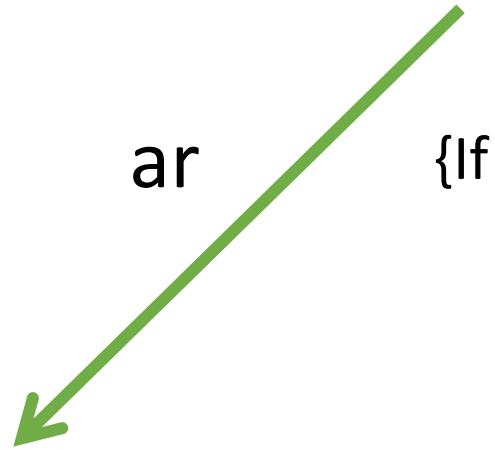
{If a is positive}

Vector Arithmetic

Multiplication by a scalar



Multiply by a;

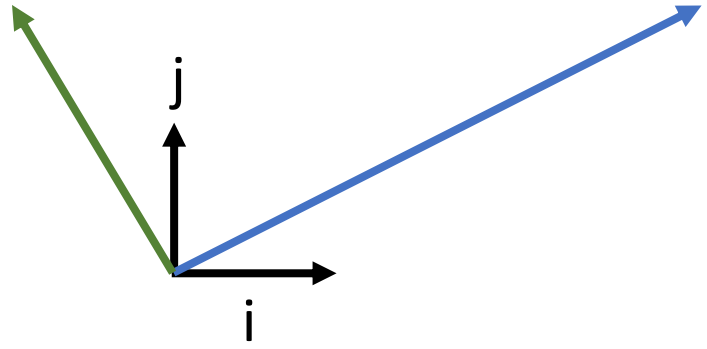


{If a is negative}

Vector Arithmetic

Modulus and Inner Product

$$\begin{bmatrix} si \\ sj \end{bmatrix} \leftarrow s = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$



$$r = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \rightarrow \begin{bmatrix} ri \\ rj \end{bmatrix}$$

Dot Product: $r \cdot s = ri\ si + rj\ sj$

$r \cdot s = 3x(-1) + 2x2$

$= 1 = s \cdot r$

Commutative

Vector Arithmetic

Transpose

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}, x \in \mathbb{R}^n \qquad x^T = [x_1 \quad x_2 \quad x_3 \quad \cdots \quad x_n], x^T \in \mathbb{R}^{1 \times n}$$

Vector Arithmetic

Transpose

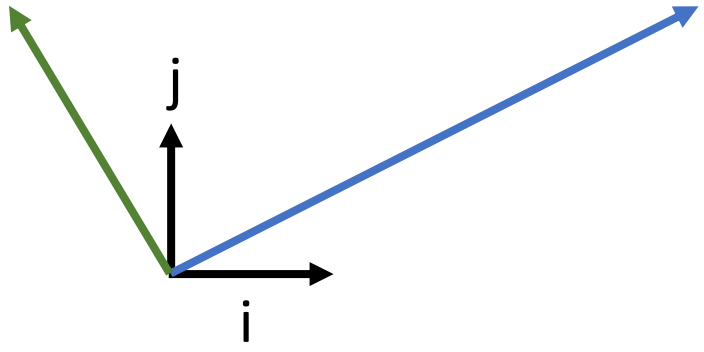
```
[16] import numpy as np
     # Make a random vector
     v = np.random.rand(5).reshape(-1, 1)
     # print vector and its shape
     print(v, v.shape)
     # transpose with T
     print(v.T, v.T.shape)
     # transpose with transpose method
     print(v.transpose(), v.transpose().shape)
     # also same as
     print(np.transpose(v), np.transpose(v).shape)
```

```
[[0.17909345]
 [0.35013963]
 [0.29005948]
 [0.56221015]
 [0.61924807]] (5, 1)
[[0.17909345 0.35013963 0.29005948 0.56221015 0.61924807]] (1, 5)
[[0.17909345 0.35013963 0.29005948 0.56221015 0.61924807]] (1, 5)
[[0.17909345 0.35013963 0.29005948 0.56221015 0.61924807]] (1, 5)
```

Vector Arithmetic

Dot Product and Transpose

$s = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$



$r = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$

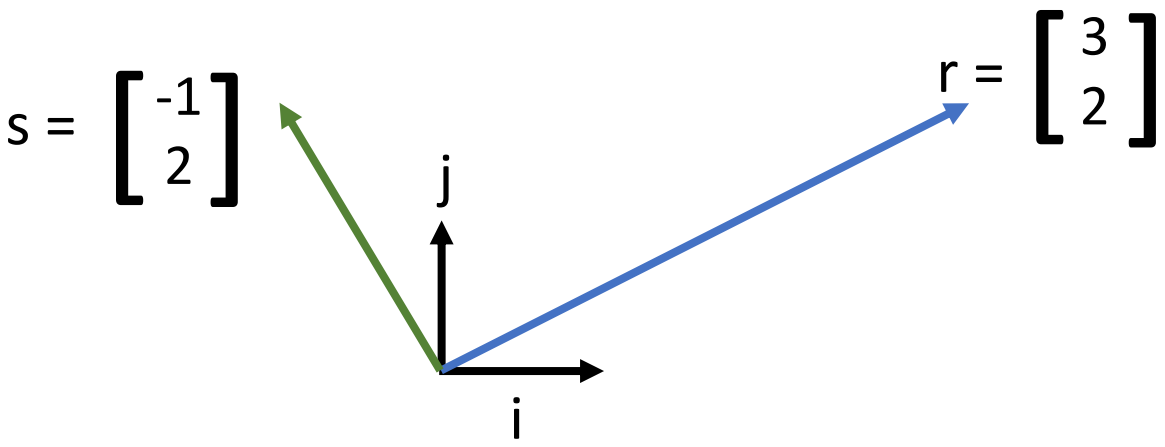
$r = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, r \in \mathbb{R}^2$

$r^T = [3 \quad 2], r^T \in \mathbb{R}^{1\times 2}$

$$r^T s = [3 \quad 2]\begin{bmatrix} -1 \\ 2 \end{bmatrix}$$
$$= (3\times -1) + (2\times 2) = 1$$
$$= r.s$$

Vector Arithmetic

Dot Product and Transpose




```
Vector s:  [-1  2]
Vector r:  [3 2]
Dot product:  1
Dot product:  1
```

```
[34] # vector dot product
     from numpy import array
     # define first vector
     s = array([-1, 2])
     print('Vector s: ', s)
     # define second vector
     r = array([3, 2])
     print('Vector r: ', r)
     # multiply vectors
     c = r.dot(s)
     print('Dot product: ', c)
     # Dot product by multiplication and transpose
     c = np.matmul(r.T, s)
     print('Dot product: ', c)
```

Vector Arithmetic

Modulus and Inner Product

Associative over scalar multiplication

$$r \cdot (as) = a(r \cdot s)$$

Vector Arithmetic

Modulus and Inner Product

Dot Product with vector itself

$$\mathbf{r} \cdot \mathbf{r} = r_1 r_1 + r_2 r_2$$

$$= r_1^2 + r_2^2$$

$$= \left(\sqrt{r_1^2 + r_2^2}\right)^2$$

$$= |\mathbf{r}|^2$$

Link between the dot product and the length or modulus of a vector

Vector Norm

- A function that ***operates on a vector (matrix)*** and returns a scalar element.
- A norm is denoted by $\mathcal{L}^{p}$ in which p ***shows the order of the norm*** and $p \geq 1, p \in \mathbb{R}$.
- The order a vector (matrix) is ***always a non-negative value***.
- The intuition behind the norm is to ***measure a kind of distance***.

Vector Norm

The l_p norm

$$l_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}, for\ p \geq 1$$

For p = 1, $\ell_1 = |x_1| + |x_2| + \ldots + |x_n|$

For p = 2, $\ell_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$

For p → ∞, $\ell_\infty = \max_i(x_1, x_2, \ldots, x_n)$

Vector Norm

The ℓ_p norm

For p = 1, $\ell_1 = |x_1|+|x_2|+\ldots+|x_n|$

```
[40] # vector L1 norm
     import numpy as np
     from numpy.linalg import norm
     # define vector
     a = np.arange(1, 6)
     print(a)
     # calculate norm
     l1 = norm(a, 1)
     print(l1)
```

```
[1 2 3 4 5]
15.0
```

Vector Norm

The ℓ_p norm

For p = 2, $\ell_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$

```
[41] # vector L2 norm
     import numpy as np
     from numpy.linalg import norm
     # define vector
     a = np.arange(1, 6)
     print(a)
     # calculate norm
     l2 = norm(a)     # or norm(a, 2)
     print(l2)
```

```
[1 2 3 4 5]
7.416198487095663
```

Vector Norm

The ℓ_p norm

For $p \rightarrow \infty$, $\ell_\infty = \max_i(x_1, x_2, \ldots, x_n)$

```
[42] # vector max norm
     from math import inf
     from numpy import array
     from numpy.linalg import norm
     # define vector
     a = np.arange(1, 6)
     print(a)
     # calculate norm
     maxnorm = norm(a, inf)
     print(maxnorm)
```

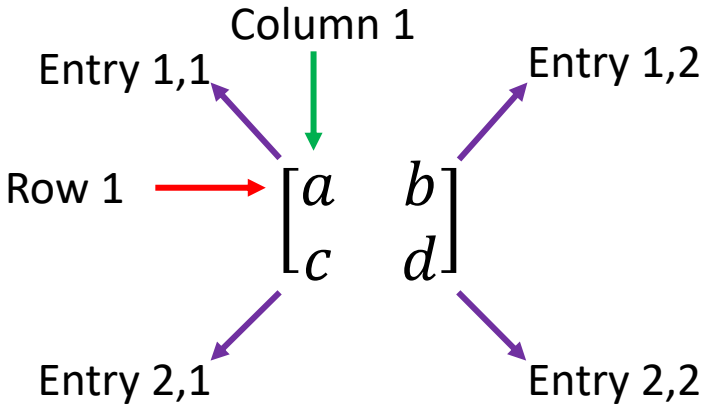
```
[1 2 3 4 5]
5.0
```

Vector Norm

ℓ_1 norm vs ℓ_2 norm

$\ell_1 = |x_1|+|x_2|+\ldots+|x_n|$, $\ell_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$

Matrices Intuition



$$A \in \mathbb{R}^{m \times n}$$

$$A = \underbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}}_{\text{n columns}} \Bigg\} \text{ m rows}$$

Matrices Notation

$$A = \begin{bmatrix} | & | & & | \\ a^1 & a^2 & \cdots & a^n \\ | & | & & | \end{bmatrix}$$

*j*th column of A by a^j or $A_{:,j}$

$$A = \begin{bmatrix} - & a_1^T & - \\ - & a_2^T & - \\ & \vdots & \\ - & a_n^T & - \end{bmatrix}$$

*i*th row of A by a^T or $A_{i,:}$

Matrices Structure

$$A = \begin{bmatrix} - & a_1^T & - \\ - & a_2^T & - \\ & \vdots & \\ - & a_n^T & - \end{bmatrix}$$

	Living area (feet^2)	Price (1 Lakh)
a_1^T ←	2104	62
a_2^T ←	1600	45
	2400	76
⋮		
	1416	80
a_5^T ←	2900	103

Matrices Structure

$$X = \begin{bmatrix} - & x_1^T & - \\ - & x_2^T & - \\ & \vdots & \\ - & x_n^T & - \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

	Living area (feet^2)	# Bedrooms	# Parks in locality	Price (1 Lakh)	
x_1^T	2104	2	1	62	y_1
x_2^T	1600	1	1	45	y_2
⋮	2400	2	0	76	⋮
	1416	1	2	80	
x_5^T	2900	3	1	103	y_5

Matrices Structure

```
[31] # Import NumPy: Numerical Python Library
     import numpy as np

     # Defining a matrix
     mat = np.array([[1, 2, 3],
                     [4, 5, 6],
                     [7, 8, 9]])

     print("Matrix: ")
     print(mat, mat.shape)
```

```
Matrix:
[[1 2 3]
 [4 5 6]
 [7 8 9]] (3, 3)
```

Matrices Arithmetic

Multiplication

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 8 & 0 \\ 3 & 4 & 7 \end{bmatrix} \times \begin{bmatrix} 0 & 2 & 3 \\ 4 & 7 & 0 \\ 3 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 17 & 28 & 21 \\ 32 & 64 & 12 \\ 37 & 62 & 51 \end{bmatrix}$$

Matrices Arithmetic

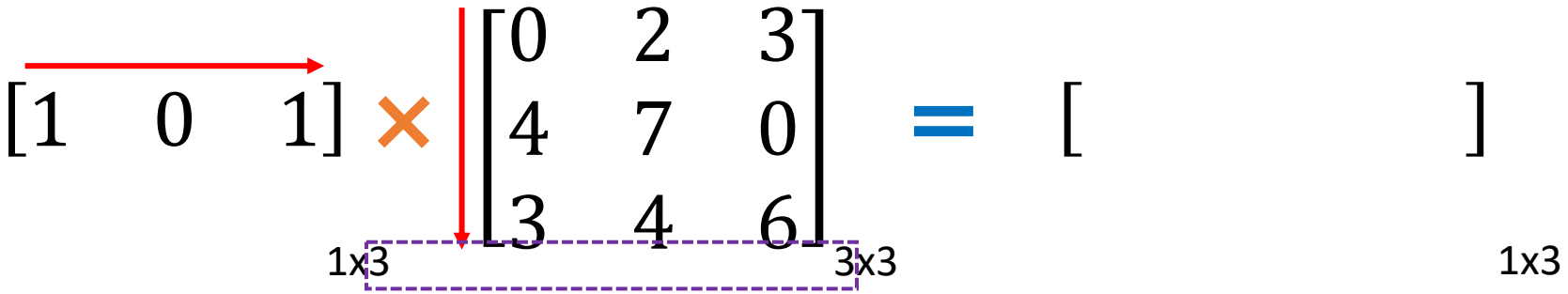
Multiplication

```
[36] # Import NumPy: Numerical Python Library
     import numpy as np

     # Defining matrices
     mat1 = np.array([[1, 2, 3],
                      [4, 8, 0],
                      [3, 4, 7]])
     print("Matrix 1: ")
     print(mat1, mat1.shape)

     mat2 = np.array([[0, 2, 3],
                      [4, 7, 0],
                      [3, 4, 6]])
     print("Matrix 2: ")
     print(mat2, mat2.shape)

     # Multiplication of two matrices
     mat3 = np.matmul(mat1, mat2)
     print("Resultant Matrix: ")
     print(mat3, mat3.shape)
```

```
Matrix 1:
[[1 2 3]
 [4 8 0]
 [3 4 7]] (3, 3)
Matrix 2:
[[0 2 3]
 [4 7 0]
 [3 4 6]] (3, 3)
Resultant Matrix:
[[17 28 21]
 [32 64 12]
 [37 62 51]] (3, 3)
```

Matrices Arithmetic

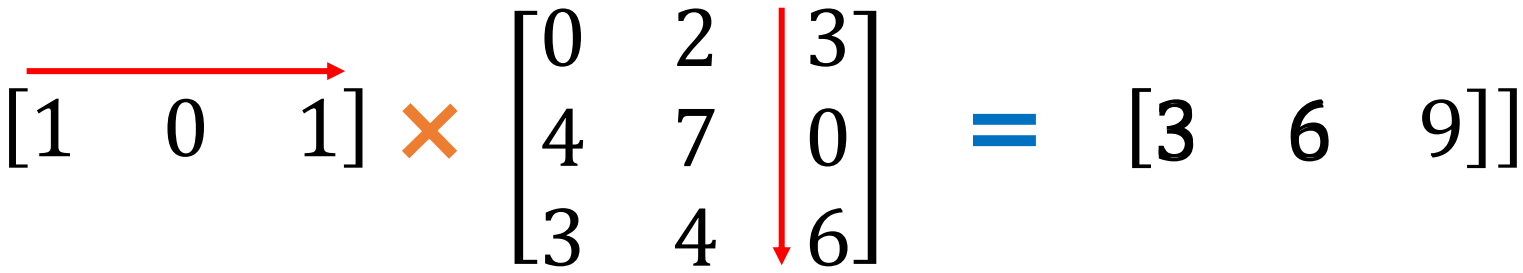
Multiplication by a vector



$$(1\times0)+(0\times4)+(1\times3)=0+0+3$$

Matrices Arithmetic

Multiplication by a vector



$$(1\times3) + (0\times0) + (1\times6) = 3 + 0 + 6$$

Matrices Arithmetic

Multiplication by a vector

```
[34] # Import NumPy: Numerical Python Library
     import numpy as np

     # Defining vector
     vec = np.array([[1, 0, 1]])
     print("Vector: ")
     print(vec, vec.shape)

     # Defining matrix
     mat = np.array([[0, 2, 3],
                     [4, 7, 0],
                     [3, 4, 6]])
     print("Matrix: ")
     print(mat, mat.shape)

     # Multiplying a row vector with matrix
     print("Resultant Matrix: ")
     print(np.matmul(vec, mat))
```

```
Vector:
[[1 0 1]] (1, 3)
Matrix:
[[0 2 3]
 [4 7 0]
 [3 4 6]] (3, 3)
Resultant Matrix:
[[3 6 9]]
```

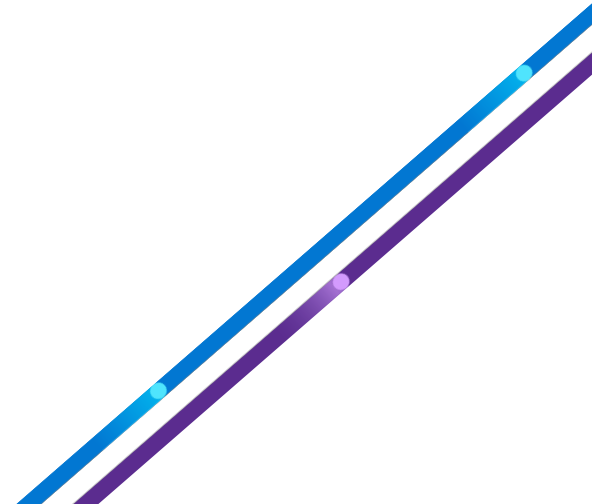
Resources to learn Python

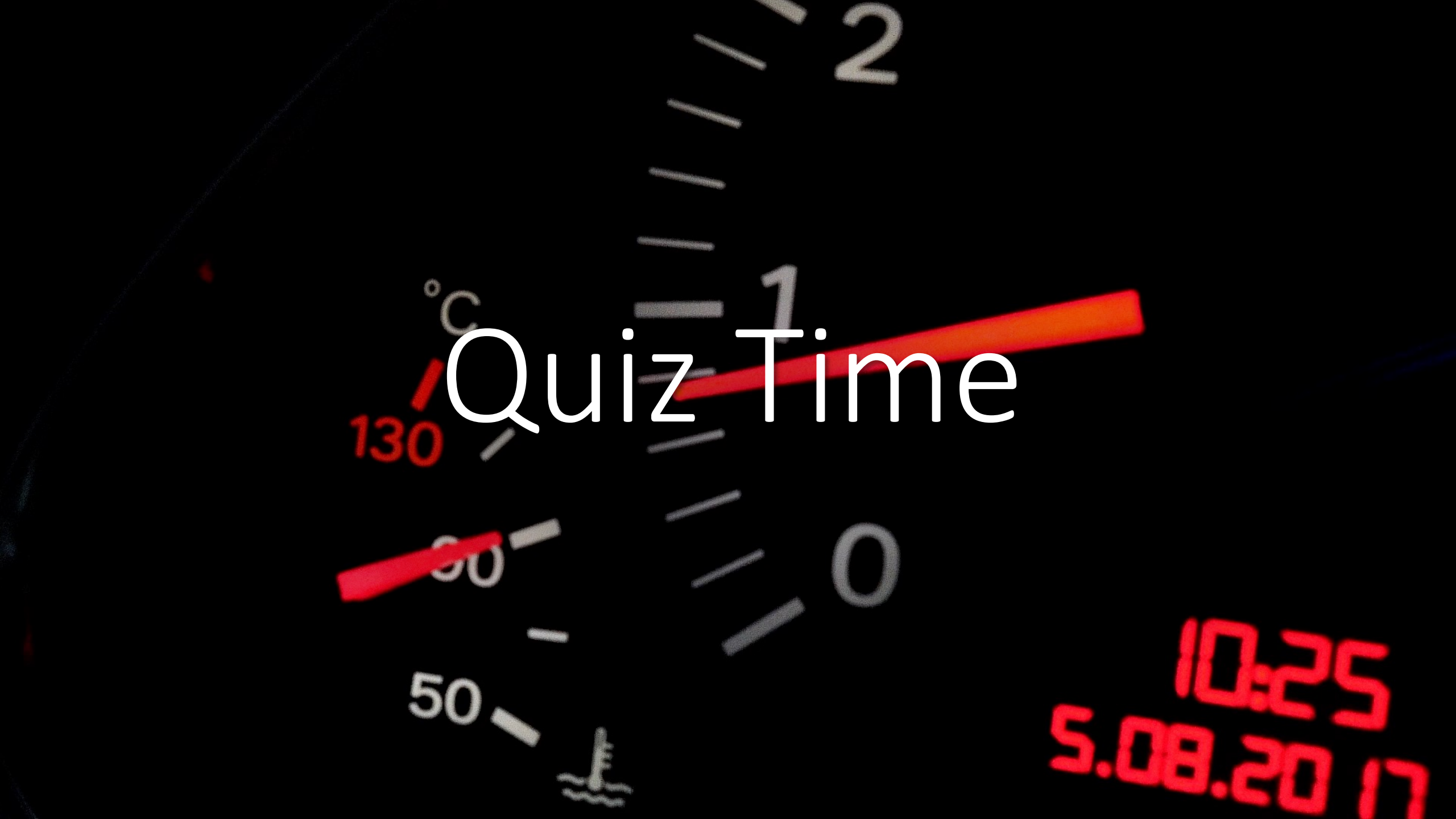
Python documentation

W3 schools (Our favorite)

CodeAcademy

Datacamp




Quiz Time