

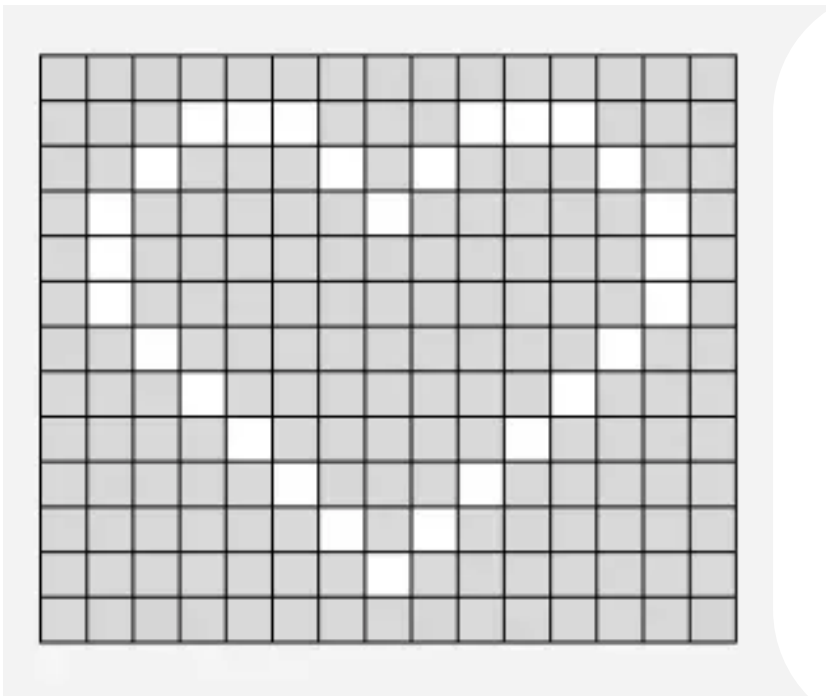
CS 1674/2074: Linear Algebra

PhD. Nils Murrugarra-Llerena

nem177@pitt.edu



Motivation: How can we encode this image?



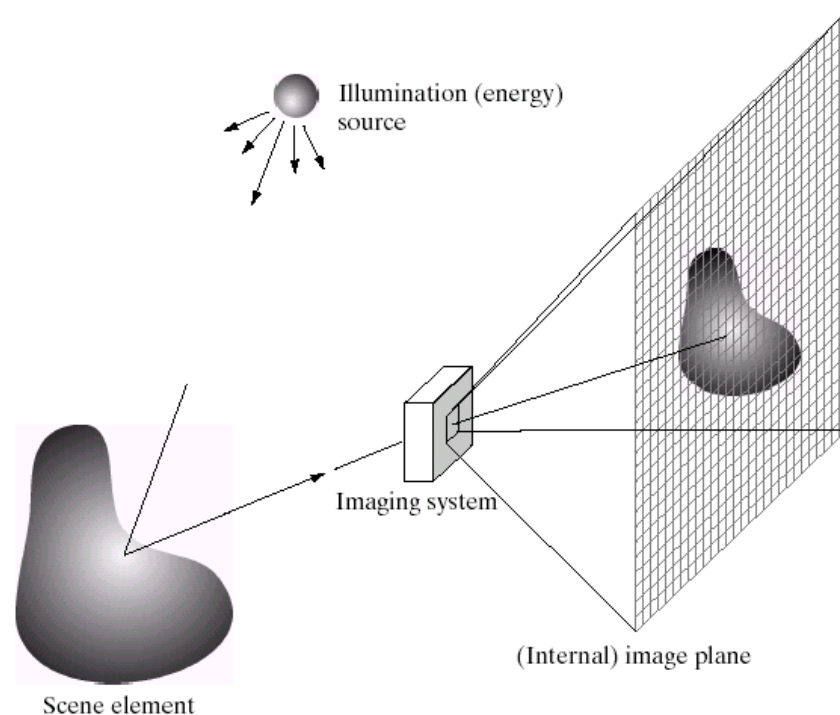
<https://www.geeksforgeeks.org/engineering-mathematics/matrices/>

Linear Algebra Review

See <http://cs229.stanford.edu/section/cs229-linalg.pdf> for more

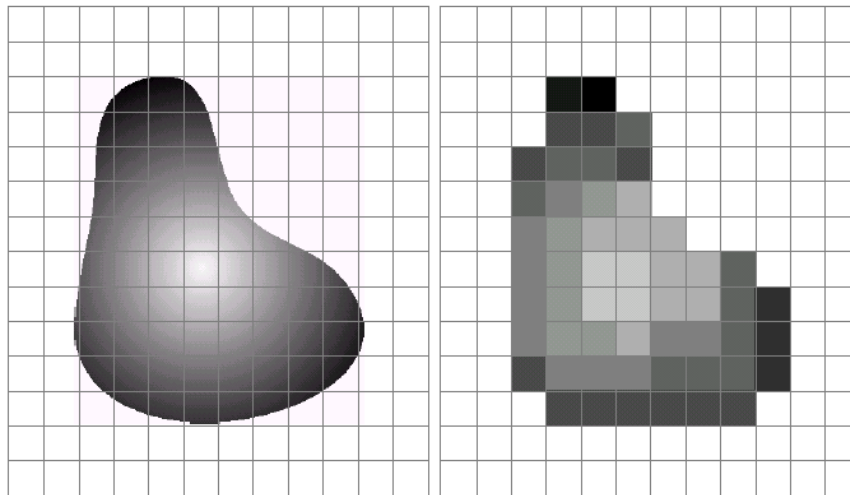
What are images? (in Python) – Image Formation

- Python treats images as matrices of numbers
- To proceed, let's talk very briefly about how images are formed



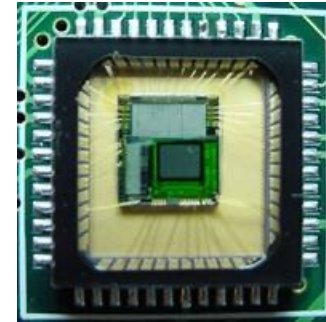
Slide credit: Derek Hoiem

Digital Images



a b

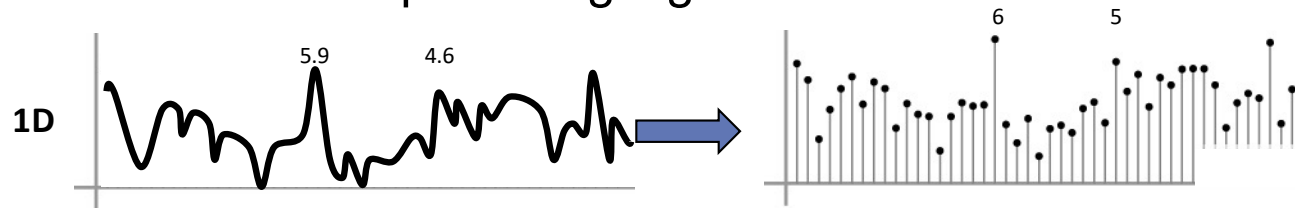
FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.



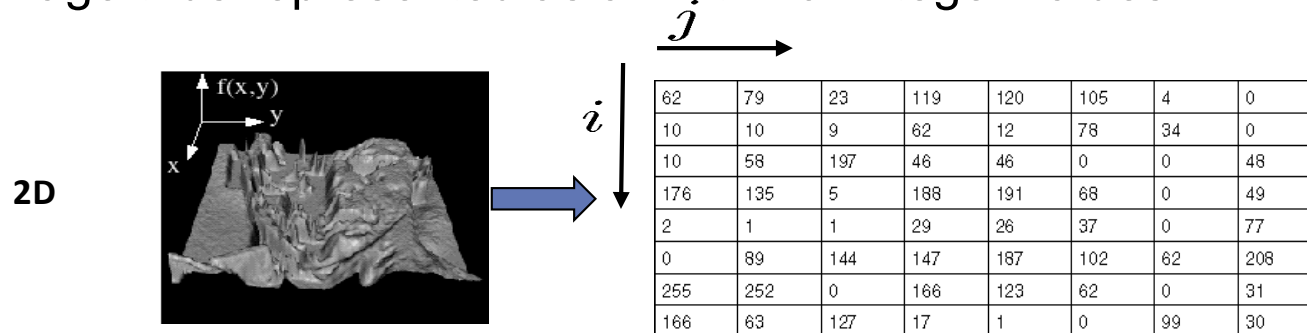
- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)

Digital Images

- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)
- What does quantizing signal look like?



- Image thus represented as a matrix of integer values.



Adapted from S. Seitz

Digital Color Images

Color images,
RGB color space:

Split image into
three channels



R



G



B

Adapted from Kristen Grauman

Images in Python

- Color images represented as a matrix with multiple channels (=1 if grayscale)
- Suppose we have a NxM RGB image called “im”
 - $im[0,0,0]$ = top-left pixel value in R-channel
 - $im(y, x, b)$ = y pixels **down**, x pixels **to right** in the b^{th} channel
 - $im(N, M, 3)$ = bottom-right pixel in B-channel
- `cv2.imread(filename)` returns a uint8 image (values **0: Black** to **255: White**)

row	column											R				G				B
		0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99								
		0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91								
		0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92	0.99							
		0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95	0.91							
		0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85	0.92							
		0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33	0.95	0.99						
		0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74	0.85	0.91						
		0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93	0.92	0.92						
		0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99	0.33	0.95						
		0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97	0.74	0.85						
		0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93	0.93	0.33						
		0.93	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93	0.99	0.74						
		0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97	0.93	0.93						
		0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93	0.99	0.99						
		0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97	0.93	0.97						
		0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93	0.93	0.93						

What is a Vector?

The Journey as a Vector

- Imagine you're planning a trip. To describe your travel, you need two key pieces of information:
 - **Direction:** The way you're going—north, south, east, or west. This tells you the path you're on.
 - **Magnitude:** The distance you're traveling—50 miles, 100 kilometers, etc. This tells you how far you're going.

A vector combines both of these ideas. It's not enough to say you traveled 100 miles [**Number/Scalar:** Distance]; you also need to specify the direction, like "100 miles due north." [**Vector:** Displacement] Without both, the information is incomplete.



Vectors and Matrices

- Vectors and matrices are just collections of ordered numbers that represent something: movements in space, scaling factors, word counts, movie ratings, pixel brightness, etc.
- We'll define some common uses and standard operations on them.



Vector

- A column vector $\mathbf{v} \in \mathbb{R}^{n \times 1}$ where

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

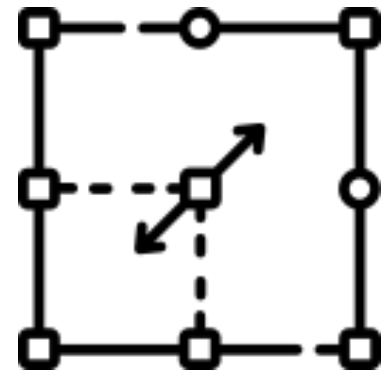
- A row vector $\mathbf{v}^T \in \mathbb{R}^{1 \times n}$ where

$$\mathbf{v}^T = [v_1 \quad v_2 \quad \dots \quad v_n]$$

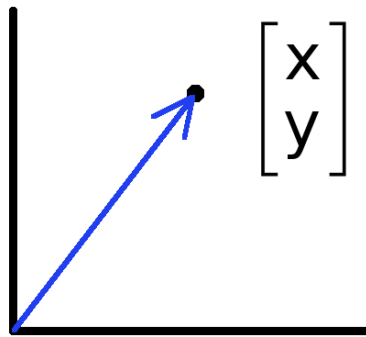
T denotes the transpose operation

Vector

- You'll want to keep track of the orientation of your vectors when programming in Python.
- You can transpose a vector V in Python by writing $V.T$.



Vectors have two main uses



- Data can also be treated as a **vector**
- Such vectors **don't have a geometric interpretation**, but **calculations like "distance" still have value**
- **Vectors** can represent an offset in 2D or 3D space
- **Points** are just vectors from the origin

Norms

- L1 norm

$$\|\mathbf{x}\|_1 := \sum_{i=1}^n |x_i|$$

- L2 norm

$$\|\mathbf{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}$$

- L^p norm (for real numbers $p \geq 1$)

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

Distances

- L1 (Manhattan) distance

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^n |p_i - q_i|$$

- L2 (Euclidean) distance

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

Example: feature representation

- A vector representing measurable characteristics of a data sample we have.
- E.g. a glass of juice can be represented via its **color** = {yellow=1, red=2, green=3, purple=4} and **taste** = {sweet=1, sour=2}
- A given glass i can be represented as a vector: $\mathbf{x}_i = [3 \ 2]$ represents **green**, **sour** juice
- For D features, this defines a D -dimensional space where we can measure similarity between samples

Example: Feature representation

L2 distance:

$$d(x1, x2) = \sqrt{4+0}$$

$$d(x1, x3) = \sqrt{0+1}$$

$$d(x2, x3) = \sqrt{4+1}$$

L1 distance:

$$d(x1, x2) = 2+0$$

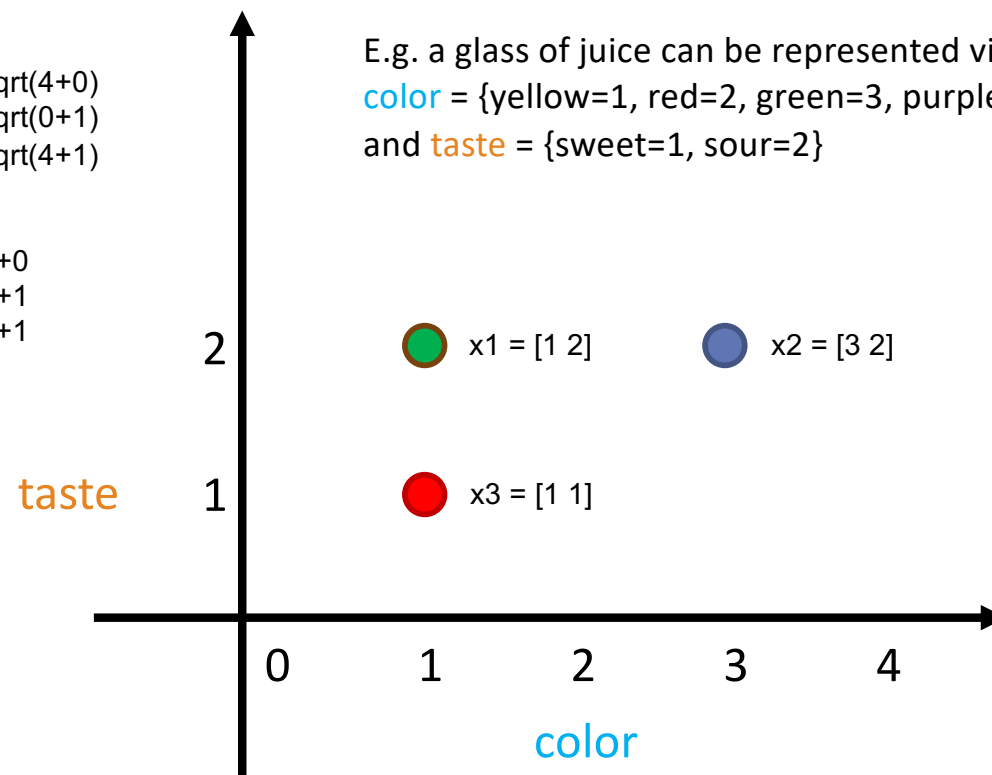
$$d(x1, x3) = 0+1$$

$$d(x2, x3) = 2+1$$

E.g. a glass of juice can be represented via its

color = {yellow=1, red=2, green=3, purple=4}

and **taste** = {sweet=1, sour=2}



Matrix

- A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is an array of numbers with size $m \downarrow$ by $n \rightarrow$, i.e. m rows and n columns.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & & & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

- If $m = n$, we say that \mathbf{A} is square.

Matrix Operations

- Addition

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} a+1 & b+2 \\ c+3 & d+4 \end{bmatrix}$$

- Can only add a matrix with matching dimensions, or a scalar.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + 7 = \begin{bmatrix} a+7 & b+7 \\ c+7 & d+7 \end{bmatrix}$$

- Scaling

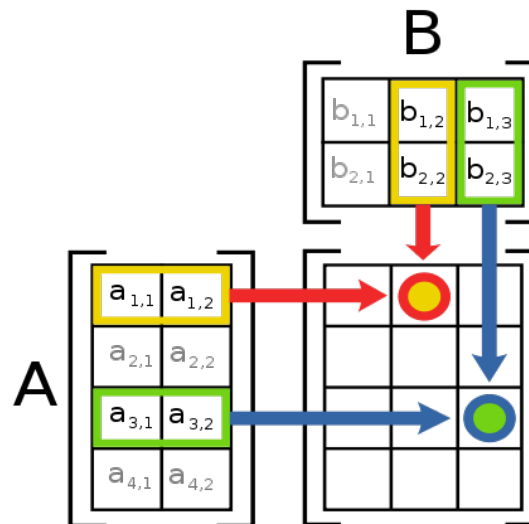
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times 3 = \begin{bmatrix} 3a & 3b \\ 3c & 3d \end{bmatrix}$$

Matrix Multiplication

- Let X be an $a \times b$ matrix, Y be an $b \times c$ matrix
- Then $Z = X * Y$ is an $a \times c$ matrix
- **Second dimension of first matrix, and first dimension of second matrix** have to be the **same**, for matrix multiplication to be possible

Matrix Multiplication

- The product AB is:



- Each entry in the result is (that row of A) dot product with (that column of B)

Matrix Multiplication

- Example:

$$\begin{array}{ccc}
 A & \times & B \\
 \downarrow & & \nearrow \\
 \begin{bmatrix} 0 & 2 \\ 4 & 6 \end{bmatrix} & & \begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix} \\
 & & \begin{bmatrix} \square & 14 \\ \square & \square \end{bmatrix}
 \end{array}$$

$$0 \cdot 3 + 2 \cdot 7 = 14$$

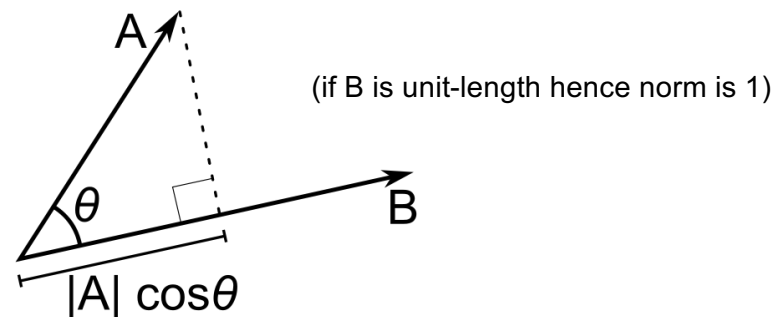
- Each entry of the matrix product is made by taking the dot product of the corresponding row in the left matrix, with the corresponding column in the right one.

Inner Product

- Multiply corresponding entries of two vectors and add up the result

$$\mathbf{x}^T \mathbf{y} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^n x_i y_i \quad (\text{scalar})$$

- $\mathbf{x} \cdot \mathbf{y}$ is also $|\mathbf{x}||\mathbf{y}|\cos(\text{angle between } \mathbf{x} \text{ and } \mathbf{y})$
- If \mathbf{B} is a unit vector, then $\mathbf{A} \cdot \mathbf{B}$ gives the length of \mathbf{A} which lies in the direction of \mathbf{B} (projection)



Different Types of Product

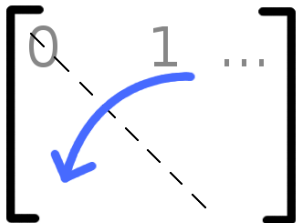
- \mathbf{x}, \mathbf{y} = column vectors ($n \times 1$)
- \mathbf{X}, \mathbf{Y} = matrices ($m \times n$)
- x, y = scalars (1×1)

- $\mathbf{x}^T \mathbf{y} = \mathbf{x} \cdot \mathbf{y}$ = inner product ($1 \times n \times n \times 1 = \text{scalar}$)
- $\mathbf{x} \otimes \mathbf{y} = \mathbf{x} \mathbf{y}^T$ = outer product ($n \times 1 \times 1 \times n = \text{matrix}$)

- $\mathbf{X}^* \mathbf{Y}$ = matrix product
- $\mathbf{X} .* \mathbf{Y}$ = element-wise product

Matrix Operations

- Transpose – flip matrix, so row 1 becomes column 1



$$\begin{bmatrix} 0 & 1 & \dots \\ 2 & 3 & \\ 4 & 5 & \end{bmatrix}^T = \begin{bmatrix} 0 & 2 & 4 \\ 1 & 3 & 5 \end{bmatrix}$$

- A useful identity:

$$(ABC)^T = C^T B^T A^T$$

Matrix Operation Properties

- Matrix addition is commutative and associative
 - $A + B = B + A$
 - $A + (B + C) = (A + B) + C$
- Matrix multiplication is associative and distributive but *not* commutative
 - $A(B * C) = (A * B)C$
 - $A(B + C) = A * B + A * C$
 - $A * B \neq B * A$

Special Matrices

- Identity matrix \mathbf{I}

- Square matrix, 1's along diagonal, 0's elsewhere
- $\mathbf{I} \cdot [\text{another matrix}] = [\text{that matrix}]$

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

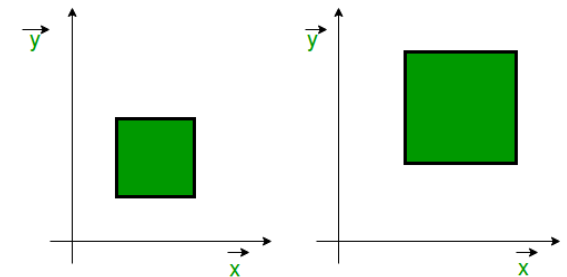
Diagonal Matrix \rightarrow Scale Matrix

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \times \begin{bmatrix} x \\ y \end{bmatrix}$$

- Diagonal matrix

- Square matrix with numbers along diagonal, 0's elsewhere
- A diagonal \cdot [another matrix] scales the rows of that matrix

$$\begin{bmatrix} 3 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 2.5 \end{bmatrix}$$



Special Matrices

- Symmetric matrix

$$\mathbf{A}^T = \mathbf{A}$$

$$\begin{bmatrix} 1 & 2 & 5 \\ 2 & 1 & 7 \\ 5 & 7 & 1 \end{bmatrix}$$

Python tutorial

Setup Conda Environment:

[CS 1674](#)

[CS 2074](#)

Python Tutorial

Duration: 60 min

https://github.com/nineil-pitt/cs1674_2074_fall25/tree/main/lab_1_python



To join, go to: ahaslides.com/BB6AU

Please, convert an image to grayscale, add your name and upload your result.

Get Feedback

Group

0 0/50