# Module 1

**Tensors and Datasets**

**Tensors 2D**

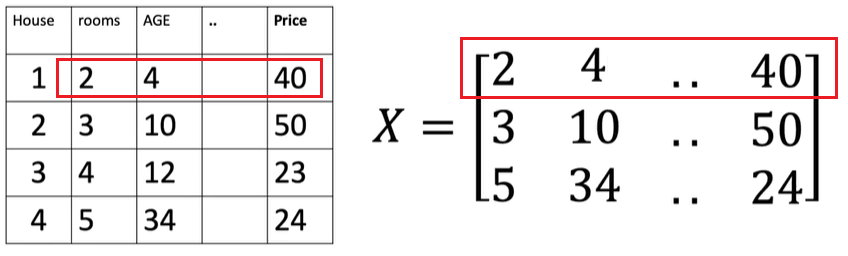
## 📌 Two-Dimensional Tensors

### 🔹 Understanding 2D Tensors

A 2D tensor is a container that holds numerical values of the same type and is typically visualized as a matrix.

In real-world applications, 2D tensors can represent:

* **Tabular data**, where each row is a sample (e.g., a house), and each column is a feature (e.g., number of rooms, age, price).



* **Grayscale images**, where pixel intensities range from 0 (black) to 255 (white), forming a 2D grid.



Tensors can be extended beyond two dimensions:

* **3D tensors** are used to represent color images, where each channel (red, green, blue) has its own 2D matrix of intensity values.



* Higher-dimensional tensors (e.g., 4D) are also used in deep learning.

### 🔹 Creating 2D Tensors

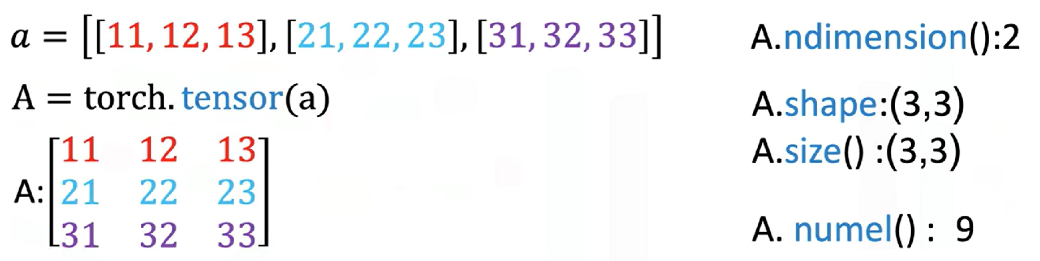
A 2D tensor can be created from a nested list, where each inner list represents a row.

The structure is interpreted as a rectangular matrix:

* The outer dimension represents rows.
* The inner dimension represents columns (eg: each element on a list, in this case since we have three elements in each list we have three columns).

Important tensor attributes:

* **Number of dimensions (rank)** can be queried to confirm the tensor structure.
* **Shape** returns the count of rows and columns.
* **Size** can be used interchangeably to obtain shape.
* **Number of elements** can be calculated by multiplying rows and columns or using a built-in method.

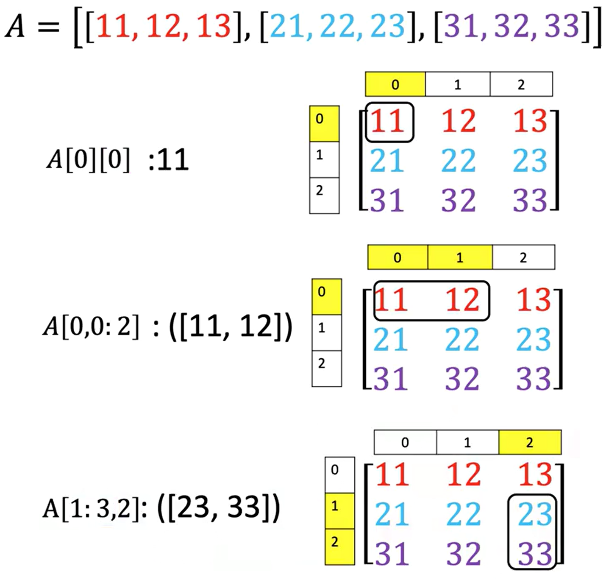
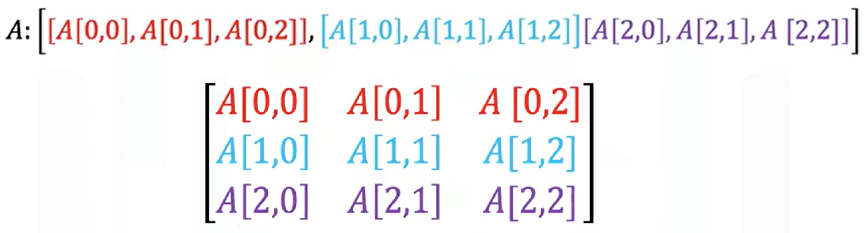


### 🔹 Indexing and Slicing

Indexing allows extraction of individual values, partial rows, or partial columns from the tensor for further computation or inspection.

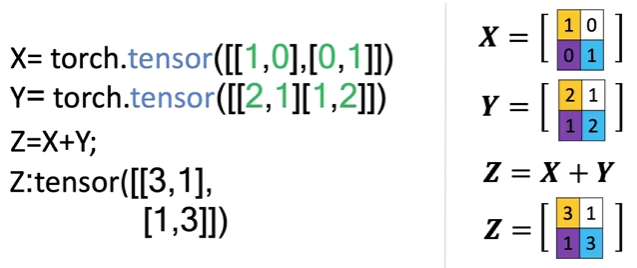
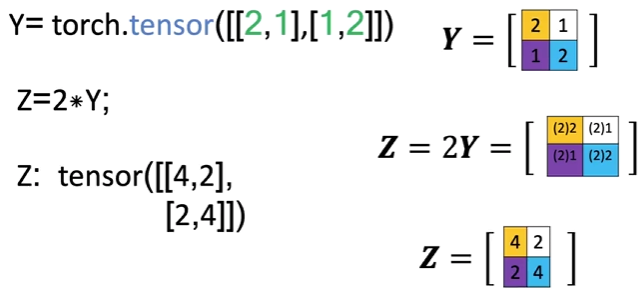
It’s performed using two indices:

* The first index corresponds to the **row**.
* The second index corresponds to the **column**.



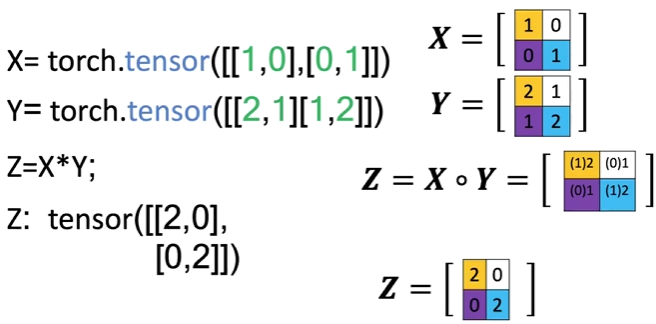
### 🔹 Basic Operations on 2D Tensors

🔸 **Addition:**

* + Two tensors of the same shape can be added together.
  + This performs element-wise addition, similar to matrix addition in linear algebra.
  + Each element in the result is the sum of the corresponding elements in the input tensors.

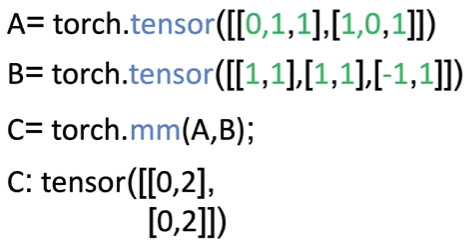
**🔸 Scalar Multiplication:**

* + Multiplying a 2D tensor by a scalar scales each individual element.
  + The resulting tensor is the same shape, but each value is multiplied by the scalar.

🔸 **Hadamard Product (Element-wise Multiplication):**

* + Multiplies corresponding elements of two tensors of the same shape.
  + Produces a new tensor where each value is the product of the matching elements from the inputs.

🔸 **Matrix Multiplication:**

* + Follows standard linear algebra rules:

The number of columns in matrix A must match the number of rows in matrix B.

* + For each element in the resulting matrix:

Compute the dot product between a row from matrix A and a column from matrix B.

* + The result is a new matrix with a shape defined by the row count of matrix A and the column count of matrix B.
  + Matrix multiplication yields a meaningful transformation of input features, often used in neural network layers.

### ✅ Takeaways

✅ 2D tensors are commonly used to represent both **structured data** (like spreadsheets or tables) and **images** (grayscale and multi-channel).

✅ Tensors can be **indexed, sliced, and reshaped** to access and manipulate specific data points or submatrices.

✅ Arithmetic operations like **addition**, **scaling**, **element-wise multiplication**, and **matrix multiplication** are supported natively and follow familiar linear algebra principles.

✅ 2D tensors provide the foundation for **layered neural network computations**, especially in the early stages of data processing and feature transformation.

✅ The structure and operations on tensors mirror real-world mathematical concepts, making them an intuitive and powerful abstraction for machine learning.