# Linear Algebra:

## What Is Algebra?

Here, we will solve this problem using techniques as they are taught today. And as a disclaimer, the reader does not need to understand each specific step to grasp the importance of this overall technique. It is my intention that the historical significance and the fact that we are able to solve the problem without any guesswork will inspire inexperienced readers to learn about these steps in greater detail. Here is the first equation again:

**x + y = 1,800**

Now, we bring in the second equation:

**⅔∙x – ½∙y = 500**

We solve this equation for y by subtracting x from each side of the equation:

y = 1,800 – x

Now, we bring in the second equation:

⅔∙x – ½∙y = 500

Since we found "1,800 – x" is equal to y, it may be substituted into the second equation:

⅔∙x – ½∙(1,800 – x) = 500

Next, distribute the negative one-half (–½) across the expression "1,800 – x":

⅔∙x + (–½∙1,800) + (–½∙–x) = 500

This simplifies to:

⅔∙x – 900 + ½∙x = 500

Add the two fractions of x together and add 900 to each side of the equation:

(7/6)∙x = 1,400

Now, divide each side of the equation by 7/6:

x = 1,200

Thus, the first field has an area of 1,200 square yards. This value may be substituted into the first equation to determine y:

(1,200) + y = 1,800

Subtract 1,200 from each side of the equation to solve for y:

y = 600

Thus, the second field has an area of 600 square yards.

Notice how often we employ the technique of doing an operation to each side of an equation. This practice is best understood as visualizing an equation as a scale with a known weight on one side and an unknown weight on the other. If we add or subtract the same amount of weight from each side, the scale remains balanced. Similarly, the scale remains balanced if we multiply or divide the weights equally.

While the technique of keeping equations balanced was almost certainly used by all civilizations to advance algebra, using it to solve this ancient Babylonian problem (as shown above) is anachronistic since this technique has only been central to algebra for the last 1,200 years.

## Linear Algebra:

Two extensions:

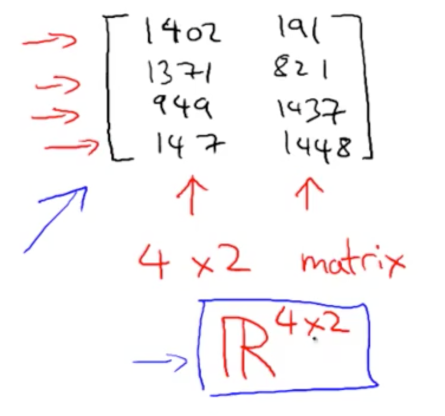
1. In min(x,y) solve for x,y exactly(algebra), without needing iterative algorithm(gradient descent). The example of x axis and y axis, plot example.
2. Learn with larger number of features:

Plot example: size(feet2) , number of bedrooms, number of floors, age of home(years) , price.

Linear algebra🡪Notation and set of the things you can do with matrics(x) and vectors(y).

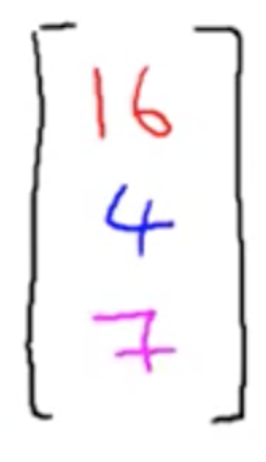
## Matrics: Rectangular array of numbers.

A Matrix is an ordered 2D array of numbers and it has two indices. The first one points to the row and the second one to the column. For example, M23 refers to the value in the second row and the third column, which is 8 in the yellow graphic above. A Matrix can have multiple numbers of rows and columns. Note that a Vector is also a Matrix, but with only one row or one column.

Means Metrix

## Vector:

Only single row or single column is called vector: like🡪 A Vector is an ordered array of numbers and can be in a row or a column. A Vector has just a single index, which can point to a specific value within the Vector.

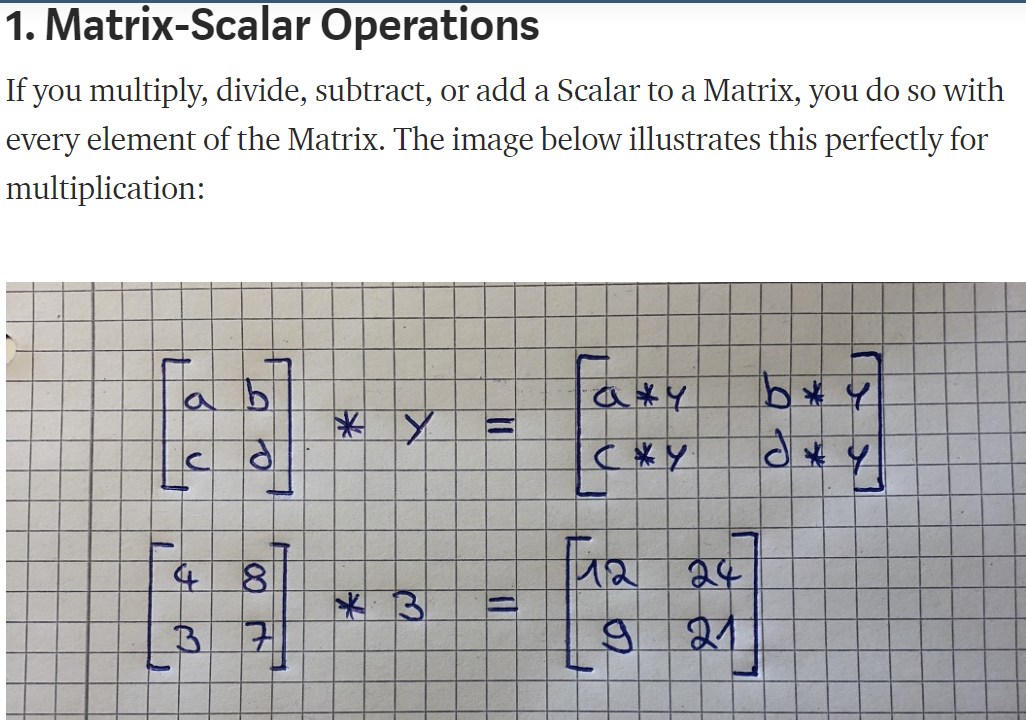


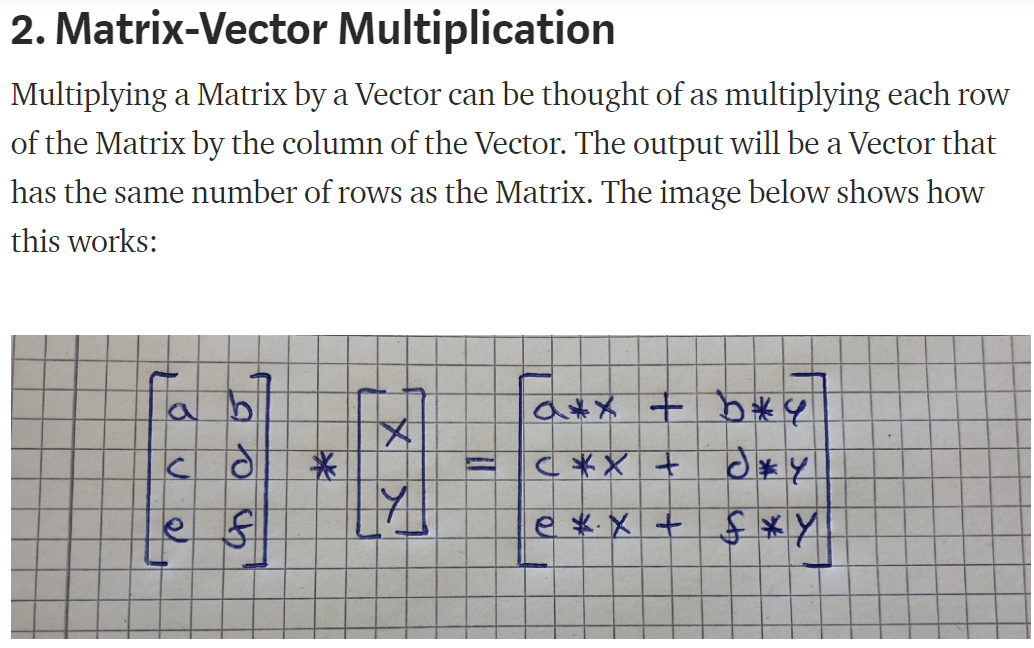
## Scalar:

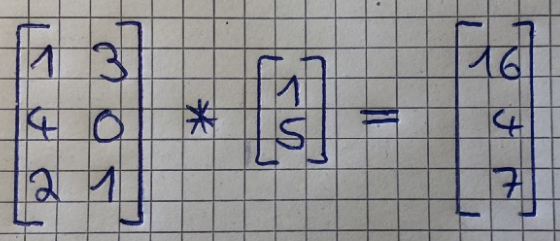
A scalar is simply a single number. For example 24.

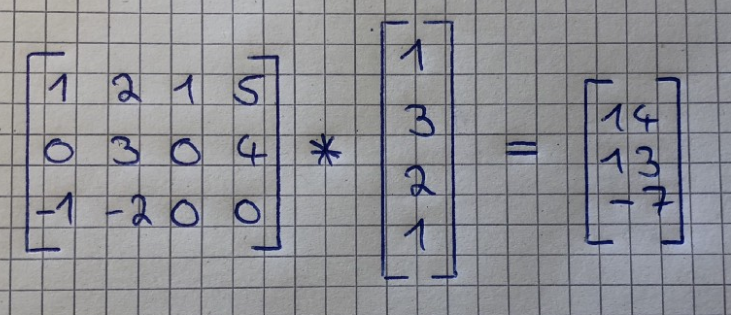


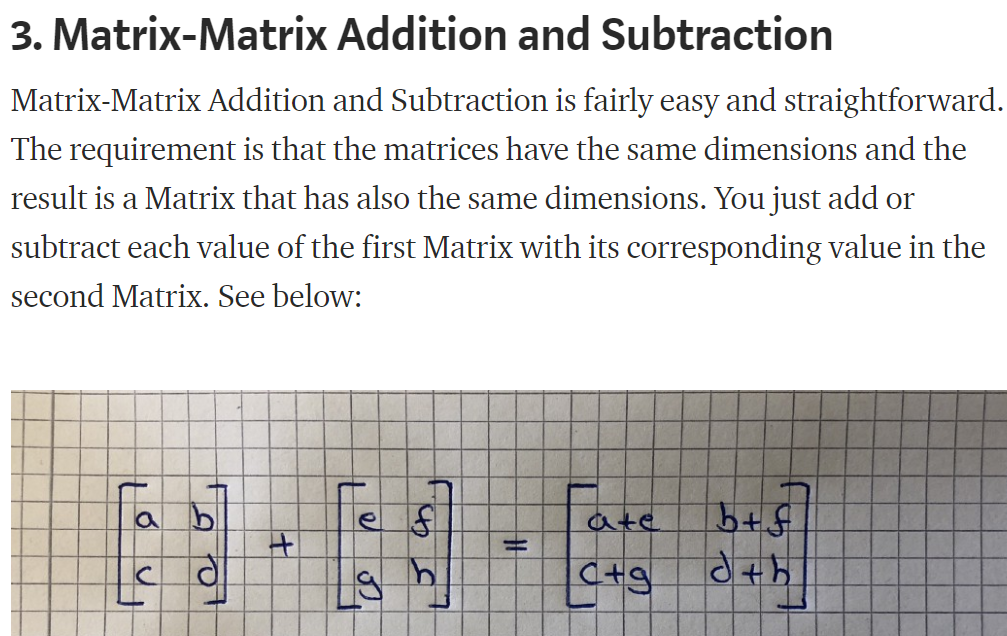
## Operations or Rules:

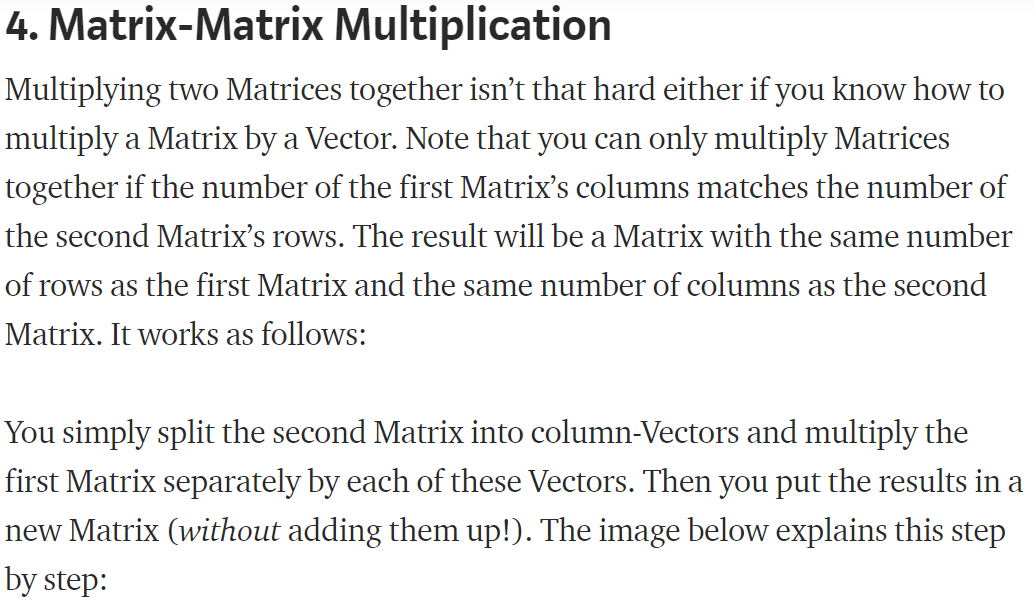


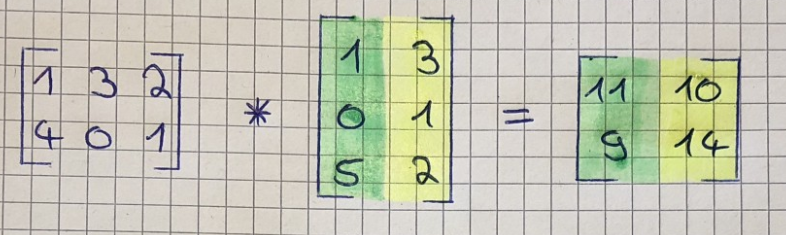


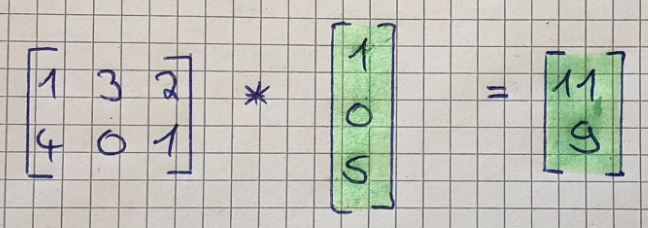


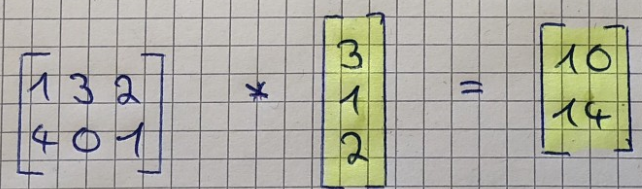


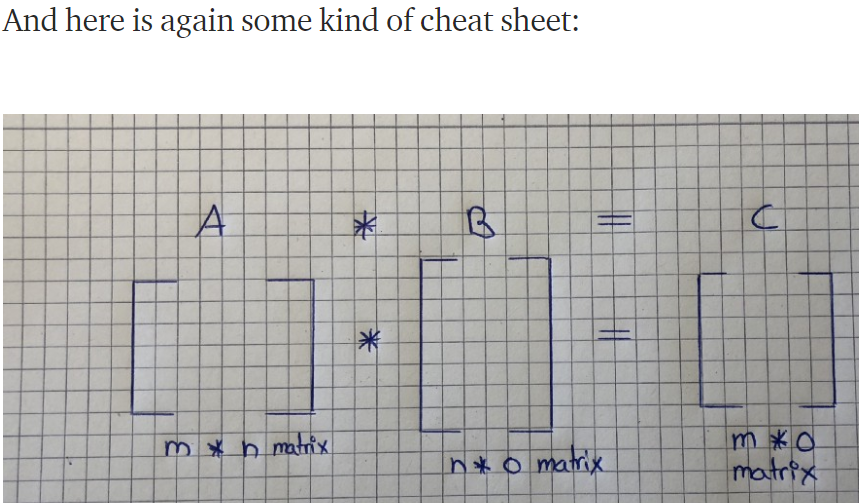




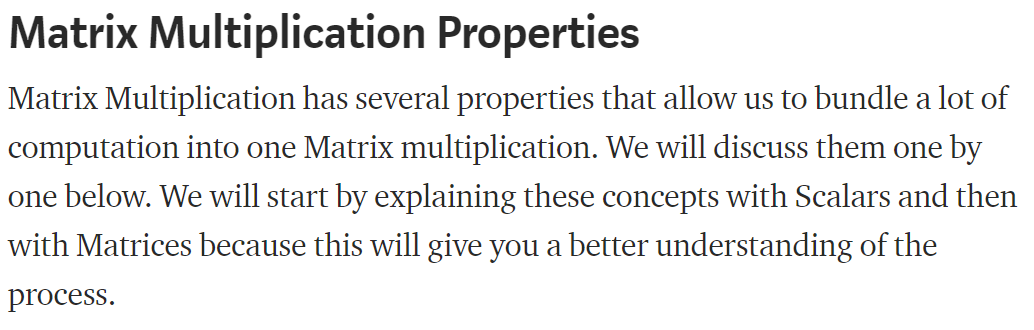






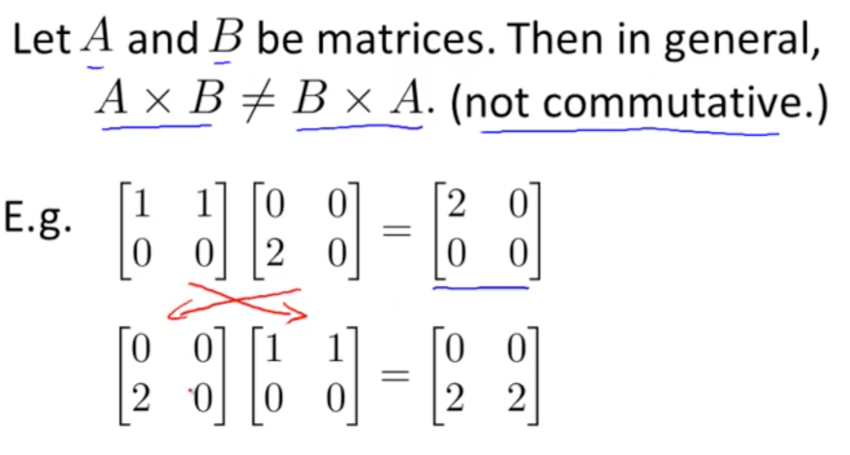


## Metrix Multiplication Properties:



### Not Commutative:

Scalar Multiplication is commutative but Matrix Multiplication is not. This means that when we are multiplying Scalars, 7\*3 is the same as 3\*7. But when we multiply Matrices by each other, A\*B isn’t the same as B\*A.



### Associative:

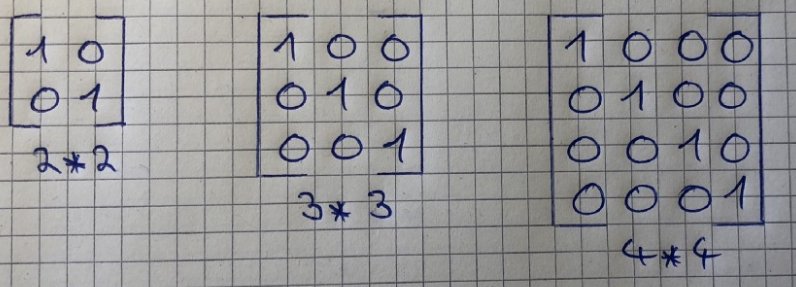
Scalar and Matrix Multiplication are both associative. This means that the Scalar multiplication 3(5\*3) is the same as (3\*5)3 and that the Matrix multiplication A(B\*C) is the same as (A\*B)C.



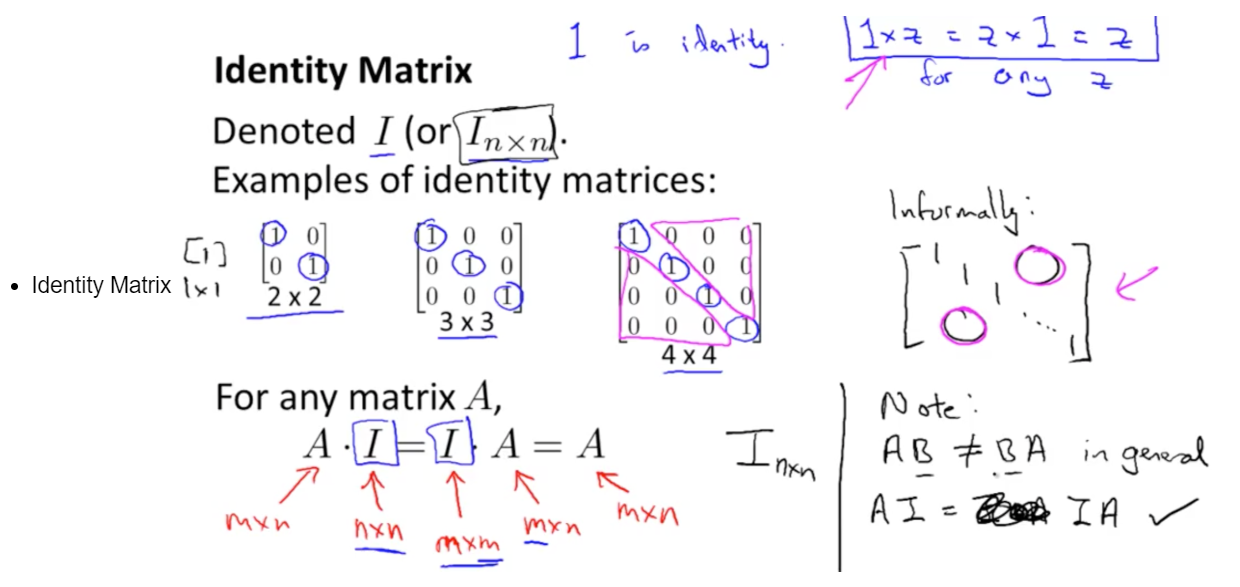
### Identity Matrix:

The Identity Matrix is a special kind of Matrix but first, we need to define what an Identity is. The number 1 is an Identity because everything you multiply with 1 is equal to itself. Therefore every Matrix that is multiplied by an Identity Matrix is equal to itself. For example, Matrix A times its Identity-Matrix is equal to A.

You can spot an Identity Matrix by the fact that it has ones along its diagonals and that every other value is zero. It is also a “squared matrix,” meaning that its number of rows matches its number of columns.



We previously discussed that Matrix multiplication is not commutative but there is one exception, namely if we multiply a Matrix by an Identity Matrix. Therefore, the following equation is true: A\*I = I\*A = A





### Distributive:

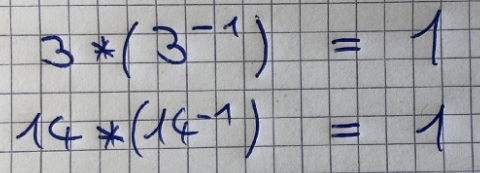
Scalar and Matrix Multiplication are also both distributive. This means that 3(5 + 3) is the same as 3\*5 + 3\*3 and that A(B+C) is the same as A\*B + A\*C.

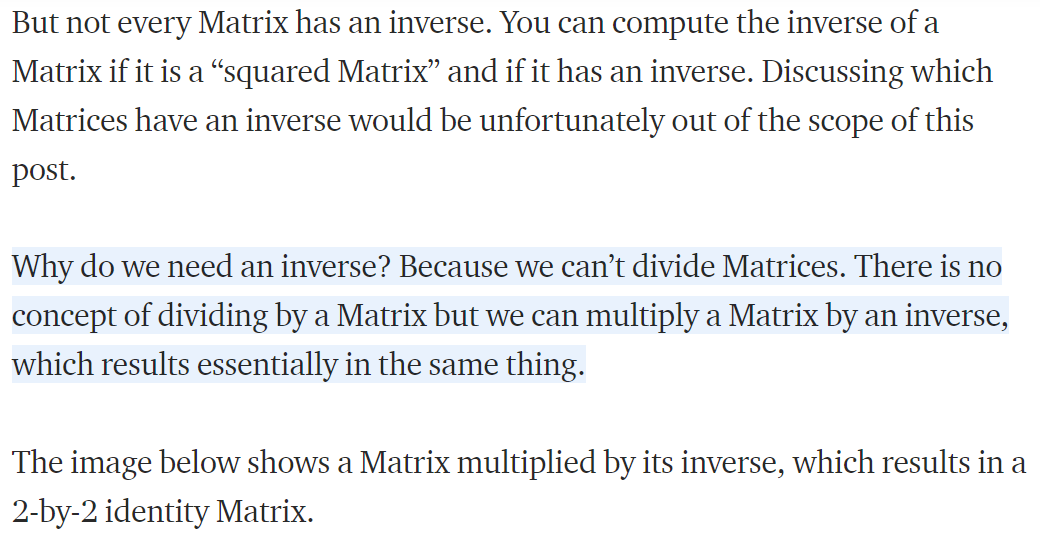
## Inverse and Transpose:

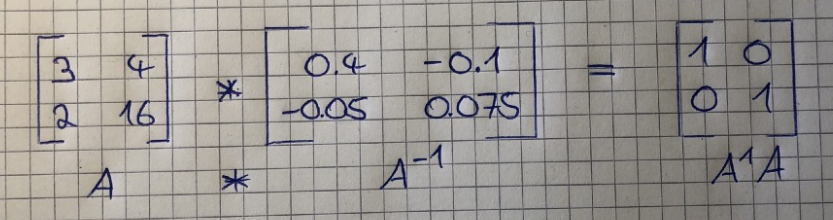
The Matrix inverse and the Matrix transpose are two special kinds of Matrix properties. Again, we will start by discussing how these properties relate to real numbers and then how they relate to Matrices.

### Inverse:

First of all, what is an inverse? A number that is multiplied by its inverse is equal to 1. Note that every number except 0 has an inverse. If you multiply a Matrix by its inverse, the result is its Identity Matrix. The example below shows what the inverse of Scalars looks like:



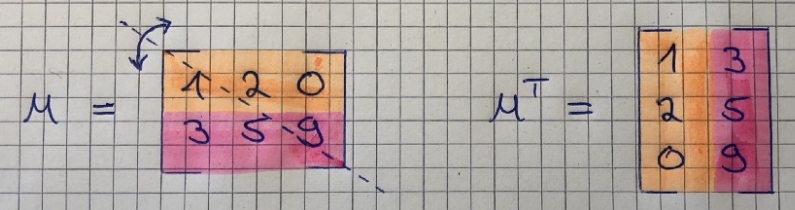


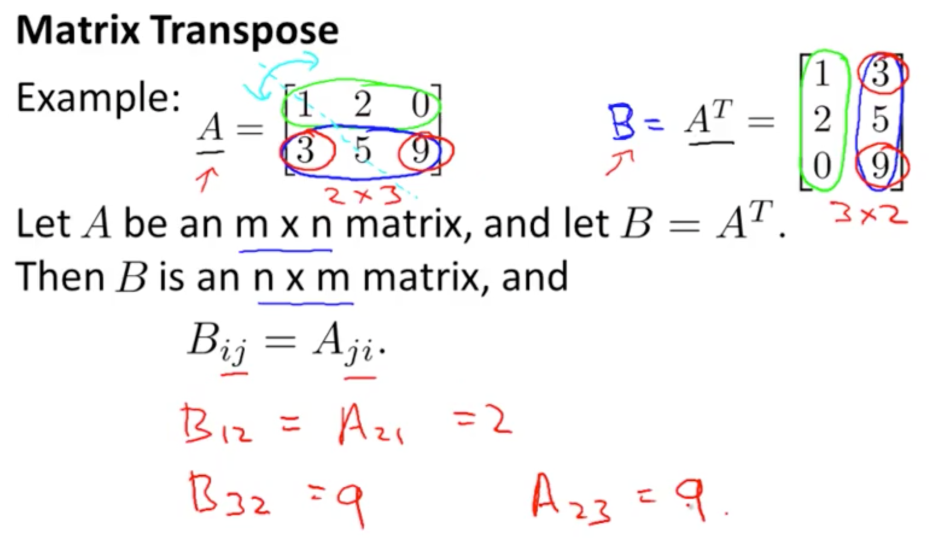




### Transpose:

And lastly, we will discuss the Matrix Transpose Property. This is basically the mirror image of a Matrix, along a 45-degree axis. It is fairly simple to get the Transpose of a Matrix. Its first column is the first row of the Matrix Transpose and the second column is the second row of the Matrix Transpose. An m\*n Matrix is transformed into an n\*m Matrix. Also, the Aij element of A is equal to the Aji(transpose) element. The image below illustrates that:





# Deep Learning:

**Definition -1:** Deep learning is a type of machine learning that mimics the neuron of the neural networks present in the human brain. Computer Vision Deep learning models are trained on a set of images a.k.a training data, to solve a task. These deep learning models are mainly used in the field of Computer Vision which allows a computer to see and visualize like a human would.

Deep learning models can be visualized as a set of points each of which makes a decision based on the inputs to the node. This sort of network is similar to the biological nervous system, with each node acting as a neuron within a larger network.

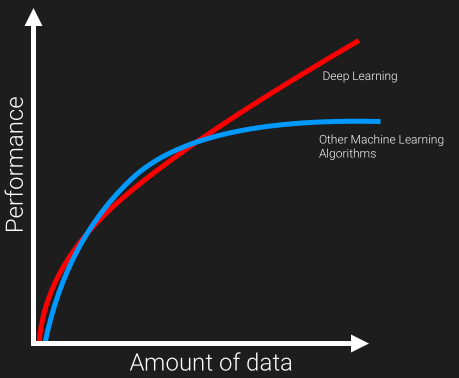
Thus, deep learning models are a class of artificial neural networks. Deep learning algorithms learn progressively about the image as it goes through each neural network layer. Early layers learn how to detect low-level features like edges, and subsequent layers combine features from earlier layers into a more holistic and complete representation.

**Definition -2:** Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

**Definition -3:** One of the machine learning technique that learns features directly from data.

## Why deep learning: When the amounth of data is increased, machine learning techniques are insufficient in terms of performance and deep learning gives better performance like accuracy.



## How deep learning differs from traditional machine learning

Unlike more traditional methods of machine learning techniques, deep learning classifiers are trained through feature learning rather than task-specific algorithms. What this means is that the machine will learn patterns in the images that it is presented with rather than requiring the human operator to define the patterns that the machine should look for in the image. The feature learning technique is used every day in how we teach a child to recognize different objects.

Feature learning is a method of traditional Machine Learning, where we use lots of features and connect all those to a basic feature.

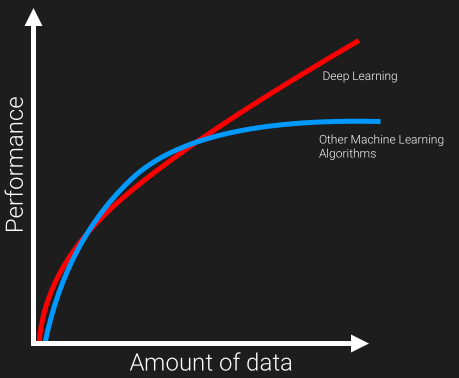
Feature learning have the freedom to be done using supervised or unsupervised type learning.

In the case of supervised feature learning, the neural network is trained using labeled input data like supervised neural networks and multilayer perceptron.

Whereas in the case of unsupervised feature learning, neural network uses unlabeled data like dictionary learning, independent component analysis, matrix factorization, it works by looking for recurring patterns.

For example, to teach a child how to identify a dog among various animals, the teacher would provide many examples of dog images, its behavior and allow the child to understand the differences between the the duo. This is feature learning at work.

The major distinguishing factor of deep learning compared to more traditional methods is the ability of the performance of the classifiers to large scaled with increased in quantities of data.



Older machine learning algorithms typically plateau in performance after it reaches a threshold of training data. Deep learning is one-of-a-kind algorithm whose performance continues to improve as more the data fed, the more the classifier is trained on resulting in outperforming more than the traditional models/ algorithm.

The execution time is comparatively more for deep learning , as it needed to be trained with lots of data. The major drawback of this ability to scale with additional training data is a need for trusted data that can be used to train the model. While the world is generating exponentially more data every year, the majority of this data is unstructured, and therefore currently unusable.

## So, what happens in Deep Learning?

The software learns, in a very realistic sense, to recognize patterns in digital representations of images, sounds, censor data and other data. We are pre-training data, in order to classify or predict and build a train/training set and test set(we know the result). And on prediction obtaining a optimal point such that our prediction gives a satisfying result.

The neurons are based out in different level and made to make their prediction at each level and most-optimal predictions, and then use the data in order to give a best-fit outcome. It is considered as true intelligence on machine.

## FAQs on Deep Learning

**What Deep Learning can do?**

1. It can also prescribe medicine used in medication.
2. Computer vision and pattern recognition
3. Robotics — Deep Learning systems have been taught to play games and even made to taught WIN games.
4. Facial recognition
5. Precision agriculture
6. Fashion technology
7. Autonomous vehicles
8. Drone and 3D mapping
9. Post estimation in Sports analytics & Retail markets
10. Security & Surveillance
11. Satellite imagery
12. Audio / Voice recognition
13. Restoring sound in videos
14. Text OCR on documents, Predicting the result of legal case a team of researchers from British and America builded a algorithm by feeding with few examples and factual information, that was able predict a court’s decision.
15. Chatbots for sales & marketing

The applications of Deep Learning and about its potential to solve real-world problems are limitless.

## Examples of Deep Learning at Work:

Deep learning applications are used in industries from automated driving to medical devices.

* Automated Driving: Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.
* Aerospace and Defense: Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.
* Medical Research: Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.
* Industrial Automation: Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.
* Electronics: Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

## How does deep learning attain such impressive results?

* In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images.
* While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:
* Deep learning requires large amounts of labeled data. For example, driverless car development requires millions of images and thousands of hours of video.
* Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

## Examples of Deep Learning at Work

* Deep learning applications are used in industries from automated driving to medical devices.
* Automated Driving: Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.
* Aerospace and Defense: Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.
* Medical Research: Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.
* Industrial Automation: Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.
* Electronics: Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

Most deep learning methods use [**neural network**](https://in.mathworks.com/discovery/neural-network.html) architectures, which is why deep learning models are often referred to as **deep neural networks**.

The term “deep” usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.

Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction.

Figure 1: Neural networks, which are organized in layers consisting of a set of interconnected nodes. Networks can have tens or hundreds of hidden layers.

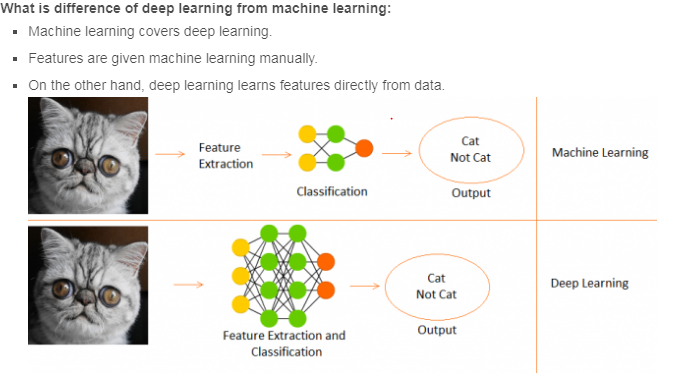
One of the most popular types of deep neural networks is known as [convolutional neural networks](https://in.mathworks.com/solutions/deep-learning/convolutional-neural-network.html)**(CNN** or**ConvNet)**. A CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images.

CNNs eliminate the need for manual [feature extraction](https://in.mathworks.com/discovery/feature-extraction.html), so you do not need to identify features used to classify images. The CNN works by extracting features directly from images. The relevant features are not pretrained; they are learned while the network trains on a collection of images. This automated feature extraction makes deep learning models highly accurate for computer vision tasks such as object classification.

Figure 2: Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image serves as the input to the next layer.

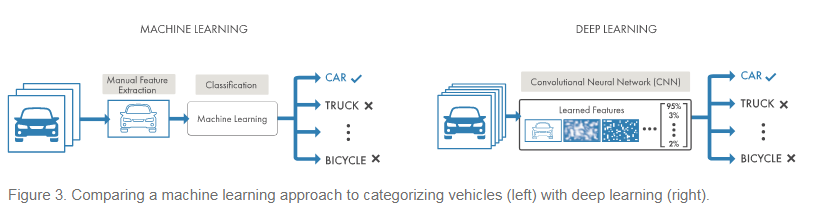
CNNs learn to detect different features of an image using tens or hundreds of hidden layers. Every hidden layer increases the complexity of the learned image features. For example, the first hidden layer could learn how to detect edges, and the last learns how to detect more complex shapes specifically catered to the shape of the object we are trying to recognize.

## What's the Difference Between Machine Learning and Deep Learning?



Deep learning is a specialized form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs “end-to-end learning” – where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically.

Another key difference is deep learning algorithms scale with data, whereas shallow learning converges. Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add more examples and training data to the network.



A key advantage of deep learning networks is that they often continue to improve as the size of your data increases.

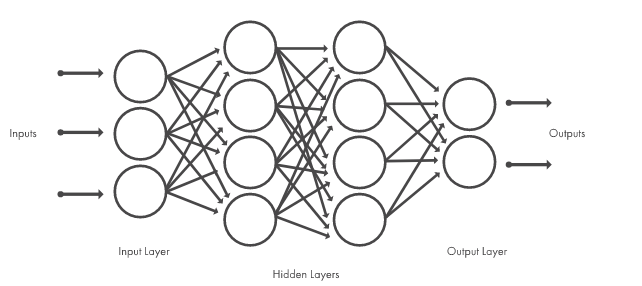


Figure 3. Comparing a machine learning approach to categorizing vehicles (left) with deep learning (right).

In machine learning, you manually choose features and a classifier to sort images. With deep learning, feature extraction and modeling steps are automatic.

## Choosing Between Machine Learning and Deep Learning

Machine learning offers a variety of techniques and models you can choose based on your application, the size of data you're processing, and the type of problem you want to solve. A successful deep learning application requires a very large amount of data (thousands of images) to train the model, as well as [GPUs, or graphics processing units](https://in.mathworks.com/solutions/gpu-computing.html), to rapidly process your data.

When choosing between machine learning and deep learning, consider whether you have a high-performance GPU and lots of labeled data. If you don’t have either of those things, it may make more sense to use machine learning instead of deep learning. Deep learning is generally more complex, so you’ll need at least a few thousand images to get reliable results. Having a high-performance GPU means the model will take less time to analyze all those images.

## How to Create and Train Deep Learning Models

The three most common ways people use deep learning to perform object classification are:

**Training from Scratch**

To train a deep network from scratch, you gather a very large labeled data set and design a network architecture that will learn the features and model. This is good for new applications, or applications that will have a large number of output categories. This is a less common approach because with the large amount of data and rate of learning, these networks typically take days or weeks to train.

**Transfer Learning**

Most deep learning applications use the [transfer learning](https://in.mathworks.com/discovery/transfer-learning.html) approach, a process that involves fine-tuning a pretrained model. You start with an existing network, such as AlexNet or GoogLeNet, and feed in new data containing previously unknown classes. After making some tweaks to the network, you can now perform a new task, such as categorizing only dogs or cats instead of 1000 different objects. This also has the advantage of needing much less data (processing thousands of images, rather than millions), so computation time drops to minutes or hours.

Transfer learning requires an interface to the internals of the pre-existing network, so it can be surgically modified and enhanced for the new task. [MATLAB®](https://in.mathworks.com/products/matlab.html) has tools and functions designed to help you do transfer learning.

**Feature Extraction**

A slightly less common, more specialized approach to deep learning is to use the network as a **feature extractor**. Since all the layers are tasked with learning certain features from images, we can pull these features out of the network at any time during the training process. These features can then be used as input to a [machine learning model](https://in.mathworks.com/solutions/machine-learning.html) such as [support vector machines (SVM)](https://in.mathworks.com/discovery/support-vector-machine.html).

## Accelerating Deep Learning Models with GPUs:

Training a deep learning model can take a long time, from days to weeks. Using GPU acceleration can speed up the process significantly. Using MATLAB with a GPU reduces the time required to train a network and can cut the training time for an image classification problem from days down to hours. In training deep learning models, MATLAB uses GPUs (when available) without requiring you to understand how to program GPUs explicitly.

Figure 4. Deep Learning Toolbox commands for training your own CNN from scratch or using a pretrained model for transfer learning.

# TensorFlow

## What is TensorFlow?

Currently, the most famous deep learning library in the world is Google's TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations.

To give a concrete example, Google users can experience a faster and more refined the search with AI. If the user types a keyword a the search bar, Google provides a recommendation about what could be the next word.

Google wants to use machine learning to take advantage of their massive datasets to give users the best experience. Three different groups use machine learning:

* Researchers
* Data scientists
* Programmers.

They can all use the same toolset to collaborate with each other and improve their efficiency.

Google does not just have any data; they have the world's most massive computer, so Tensor Flow was built to scale. TensorFlow is a library developed by the Google Brain Team to accelerate machine learning and deep neural network research.

It was built to run on multiple CPUs or GPUs and even mobile operating systems, and it has several wrappers in several languages like Python, C++ or Java.

## History of TensorFlow

A couple of years ago, deep learning started to outperform all other machine learning algorithms when giving a massive amount of data. Google saw it could use these deep neural networks to improve its services:

* Gmail
* Photo
* Google search engine

They build a framework called **Tensorflow** to let researchers and developers work together on an AI model. Once developed and scaled, it allows lots of people to use it.

It was first made public in late 2015, while the first stable version appeared in 2017. It is open source under Apache Open Source license. You can use it, modify it and redistribute the modified version for a fee without paying anything to Google.

## TensorFlow Architecture:

Tensorflow architecture works in three parts:

* Preprocessing the data
* Build the model
* Train and estimate the model

It is called Tensorflow because it takes input as a multi-dimensional array, also known as **tensors**. You can construct a sort of **flowchart** of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

## Where can Tensorflow run?

TensorFlow can hardware, and software requirements can be classified into

Development Phase: This is when you train the mode. Training is usually done on your Desktop or laptop.

Run Phase or Inference Phase: Once training is done Tensorflow can be run on many different platforms. You can run it on

* Desktop running Windows, macOS or Linux
* Cloud as a web service
* Mobile devices like iOS and Android

You can train it on multiple machines then you can run it on a different machine, once you have the trained model.

The model can be trained and used on GPUs as well as CPUs. GPUs were initially designed for video games. In late 2010, Stanford researchers found that GPU was also very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix multiplication. TensorFlow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, TensorFlow can be accessed and controlled by other languages mainly, Python.

Finally, a significant feature of TensorFlow is the TensorBoard. The TensorBoard enables to monitor graphically and visually what TensorFlow is doing.

## Introduction to Components of TensorFlow:

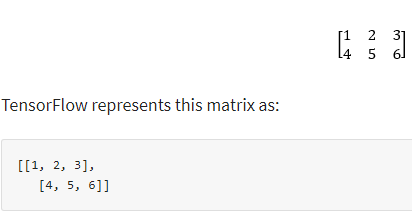
### Tensor**:**

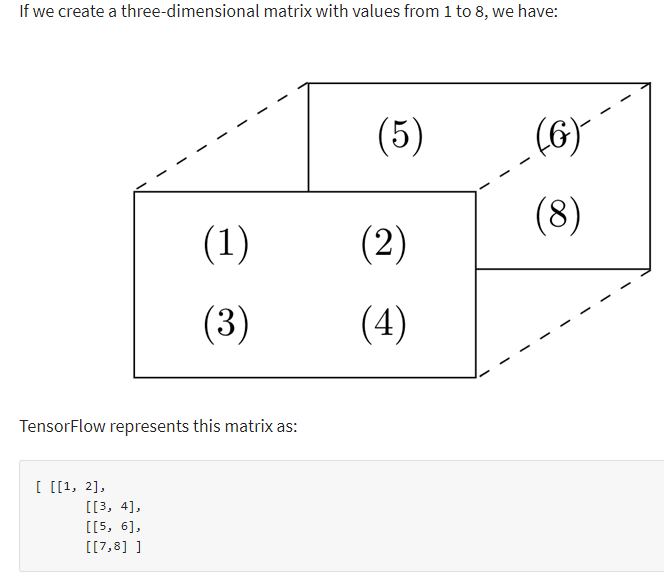
Tensorflow's name is directly derived from its core framework: **Tensor**. In Tensorflow, all the computations involve tensors. A tensor is a **vector** or **matrix** of n-dimensions that represents all types of data. All values in a tensor hold identical data type with a known (or partially known) **shape**. The shape of the data is the dimensionality of the matrix or array.

A tensor can be originated from the input data or the result of a computation. In TensorFlow, all the operations are conducted inside a **graph**. The graph is a set of computation that takes place successively. Each operation is called an **op node** and are connected to each other.

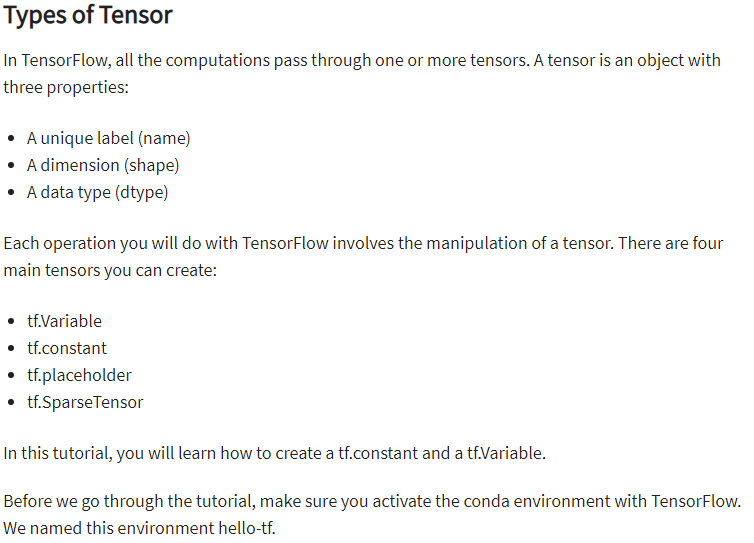
The graph outlines the ops and connections between the nodes. However, it does not display the values. The edge of the nodes is the tensor, i.e., a way to populate the operation with data.

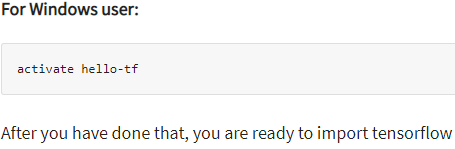
#### Representation of a Tensor: In TensorFlow, a tensor is a collection of feature vectors (i.e., array) of n-dimensions. For instance, if we have a 2x3 matrix with values from 1 to 6, we write:



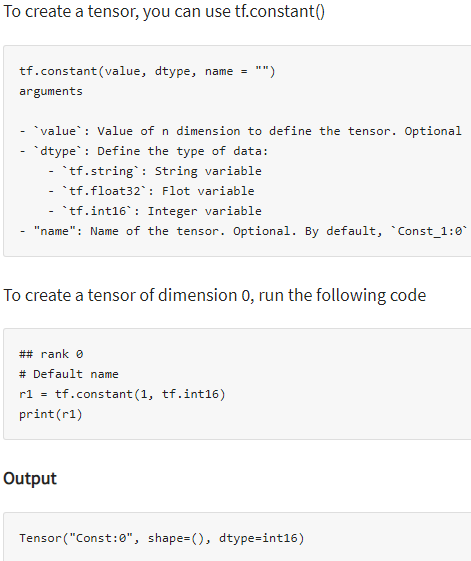


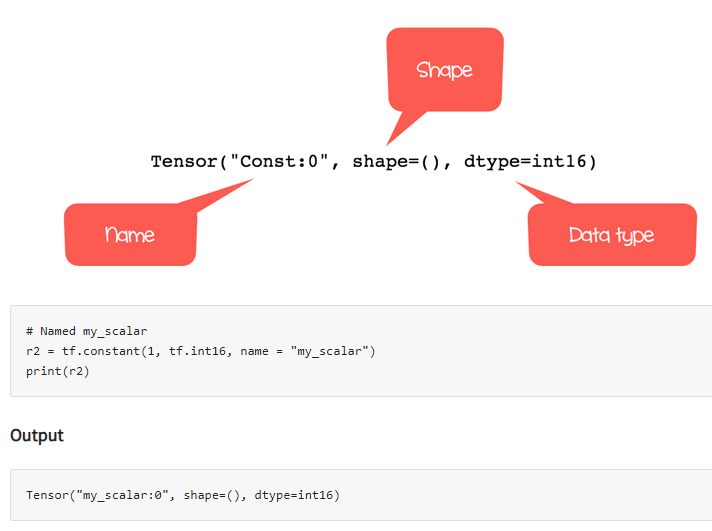
**Note**: A tensor can be represented with a scalar or can have a shape of more than three dimensions. It is just more complicated to visualize higher dimension level.





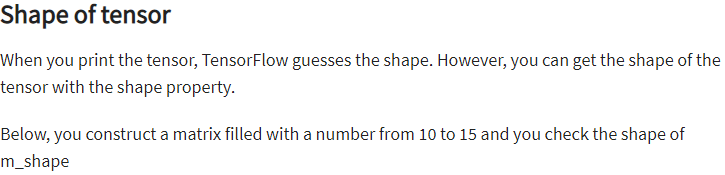


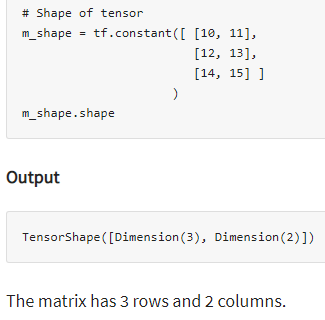


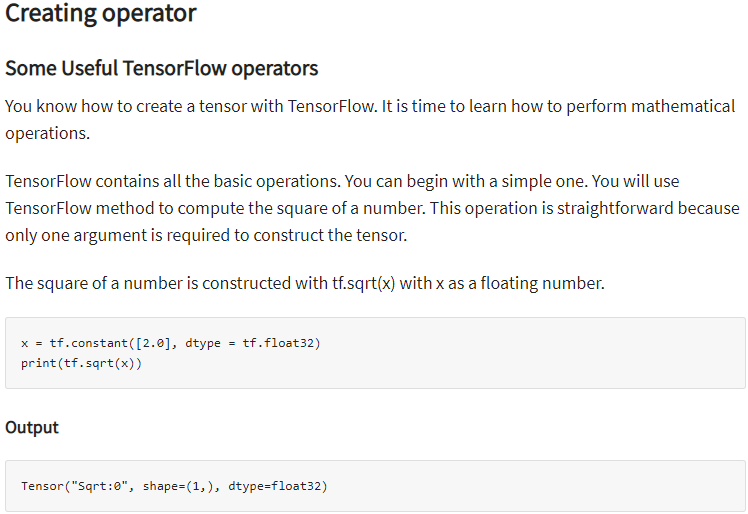


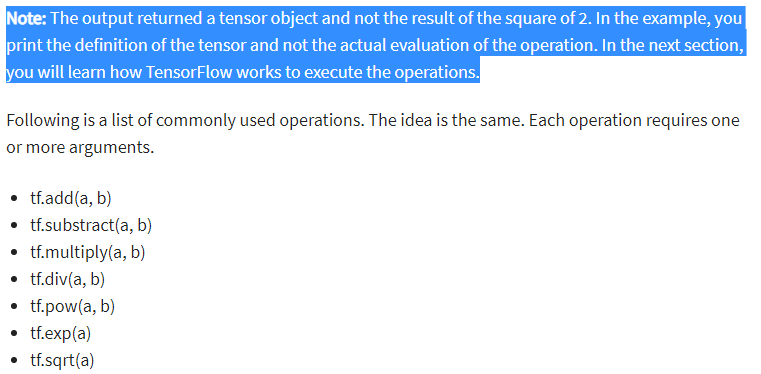


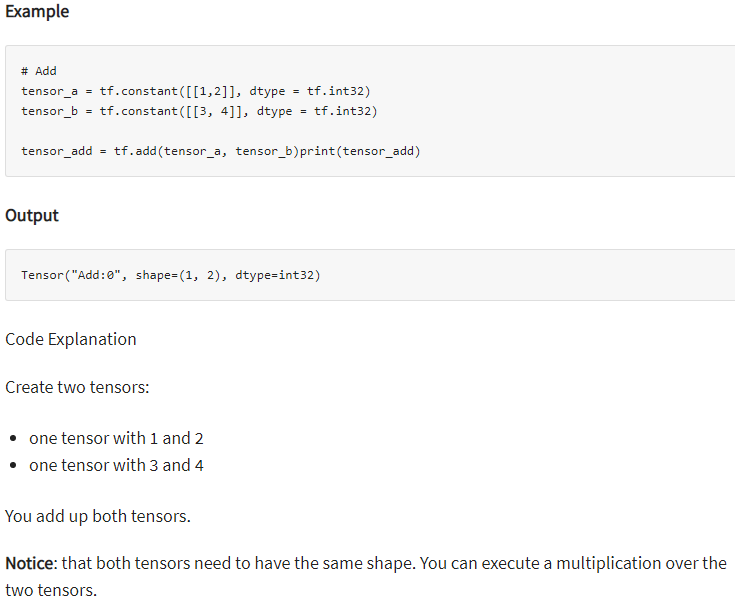


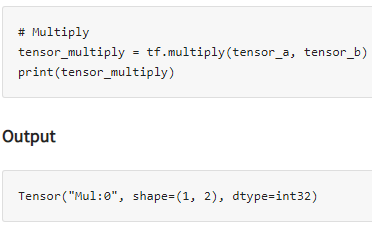




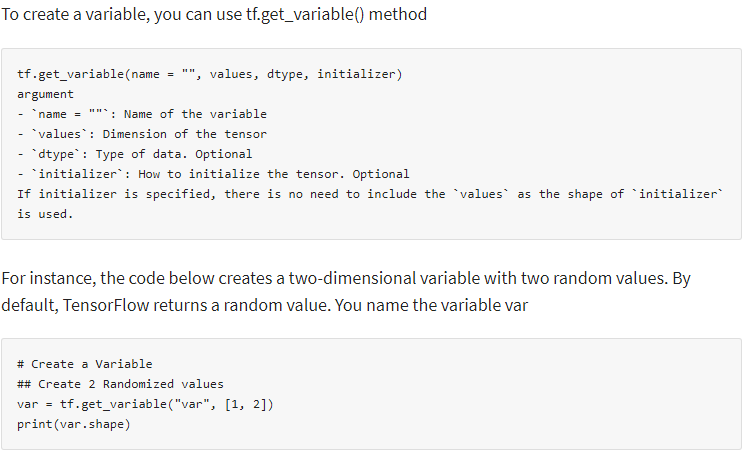


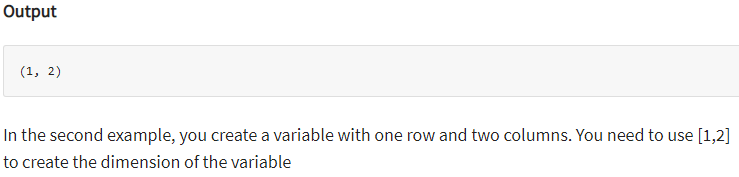


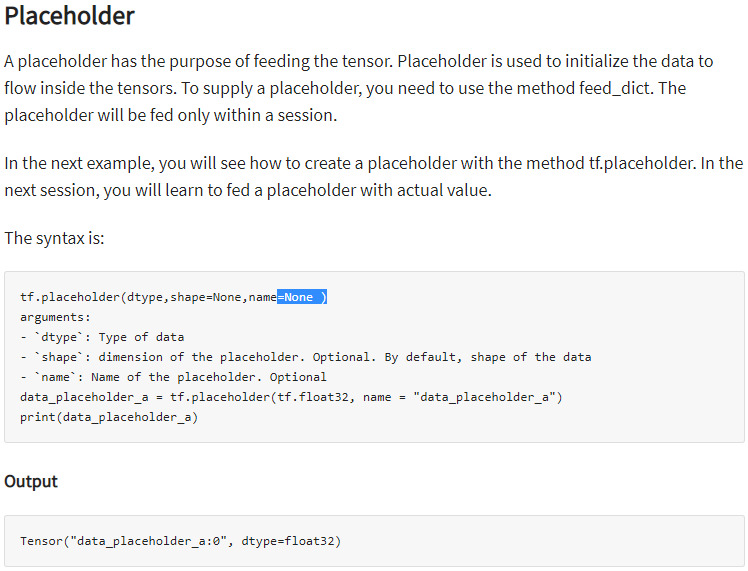




### Variables: So far, you have only created constant tensors. It is not of great use. Data always arrive with different values, to capture this, you can use the Variable class. It will represent a node where the values always change.

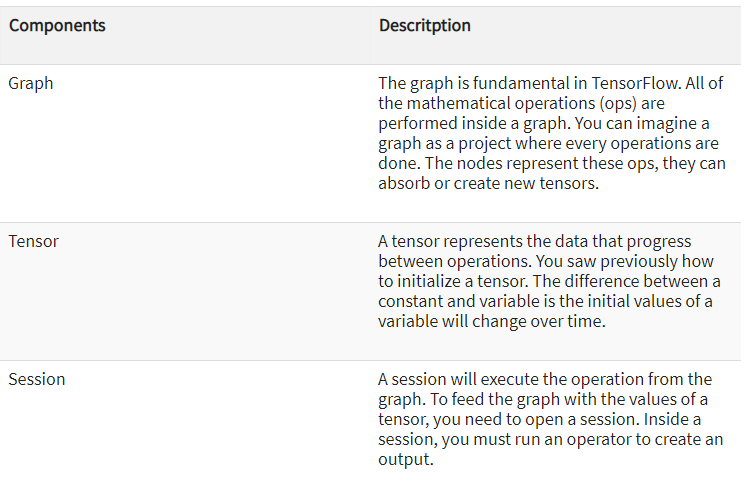


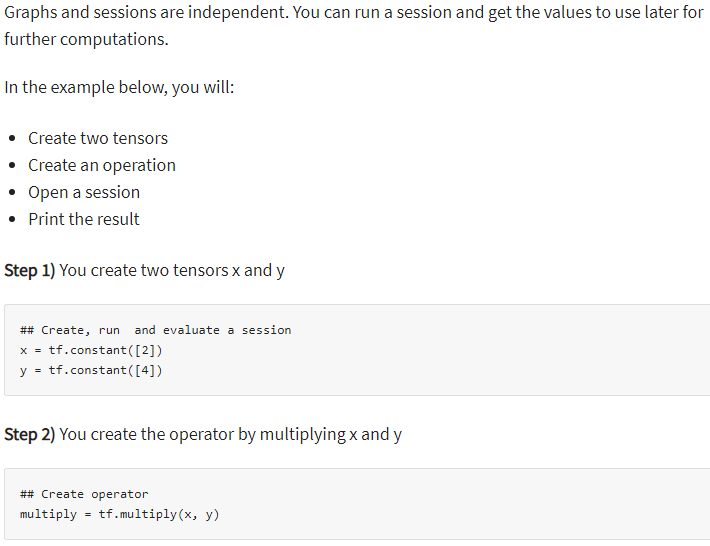


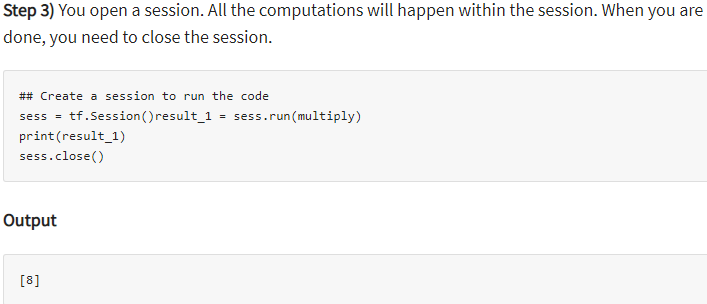


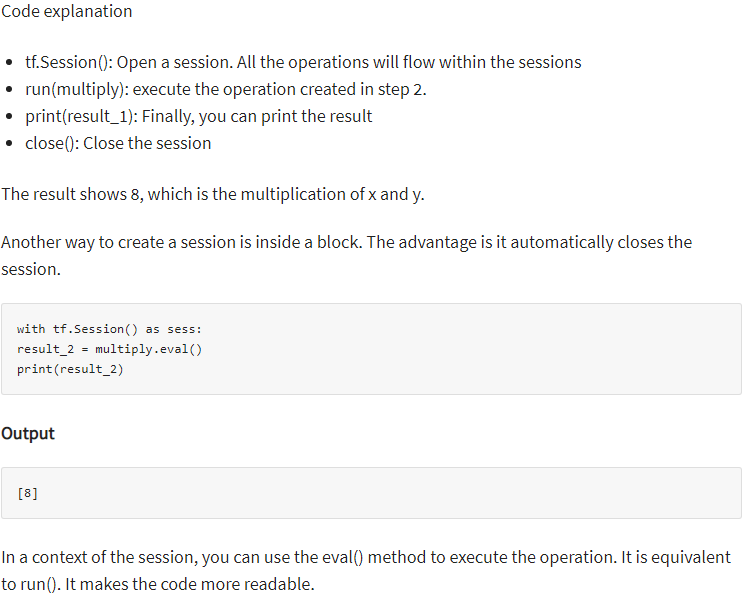
## Session: TensorFlow works around 3 main components:

* Graph
* Tensor
* Session









### ****Graphs:****

TensorFlow makes use of a graph framework. The graph gathers and describes all the series computations done during the training. The graph has lots of advantages:

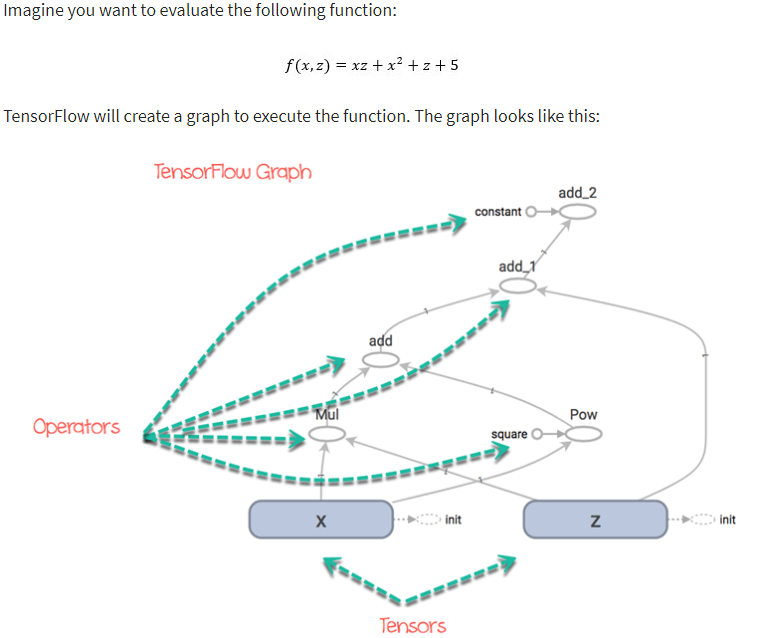
* It was done to run on multiple CPUs or GPUs and even mobile operating system
* The portability of the graph allows to preserve the computations for immediate or later use. The graph can be saved to be executed in the future.
* All the computations in the graph are done by connecting tensors together
  + A tensor has a node and an edge. The node carries the mathematical operation and produces an endpoints outputs. The edges the edges explain the input/output relationships between nodes.

TensorFlow depends on a genius approach to render the operation. All the computations are represented with a dataflow scheme. The dataflow graph has been developed to see to data dependencies between individual operation. Mathematical formula or algorithm are made of a number of successive operations. A graph is a convenient way to visualize how the computations are coordinated.

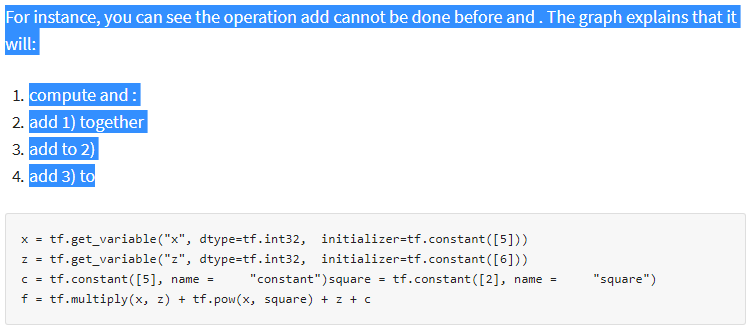
The graph shows a node and an edge. The node is the representation of a operation, i.e. the unit of computation. The edge is the tensor, it can produce a new tensor or consume the input data. It depends on the dependencies between individual operation.

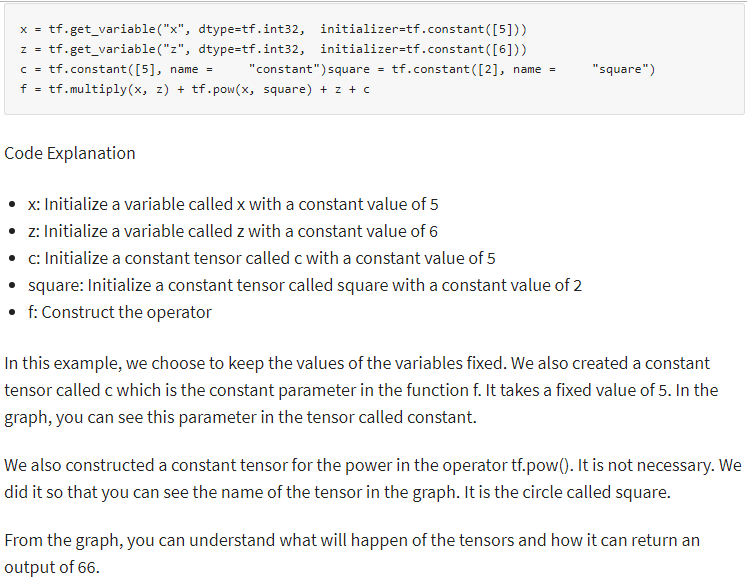
The structure of the graph connects together the operations (i.e. the nodes) and how those are operation are feed. Note that the graph does not display the output of the operations, it only helps to visualize the connection between individual operations.

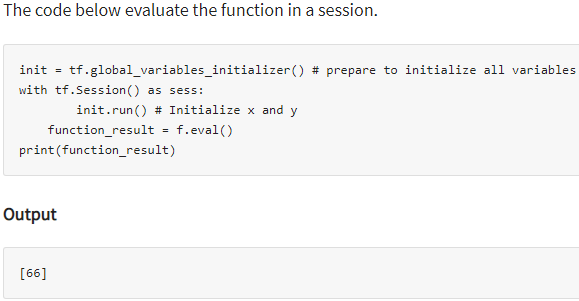
Let's see an example :

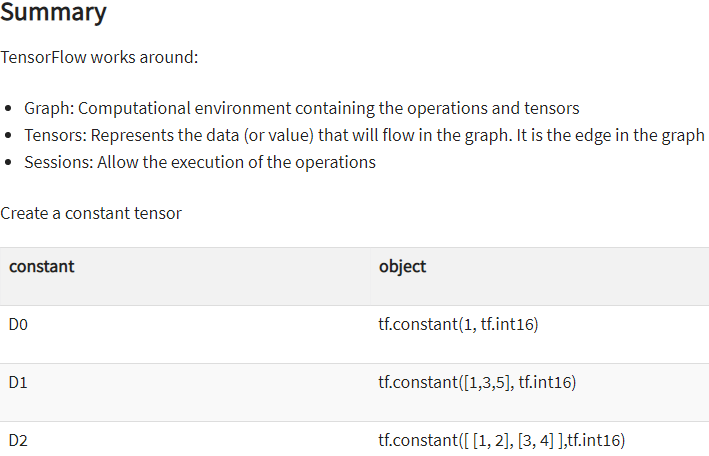


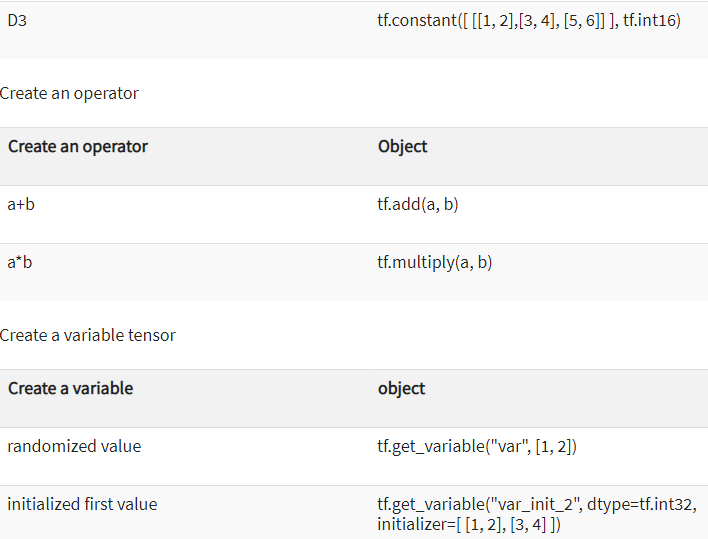
You can easily see the path that the tensors will take to reach the final destination.

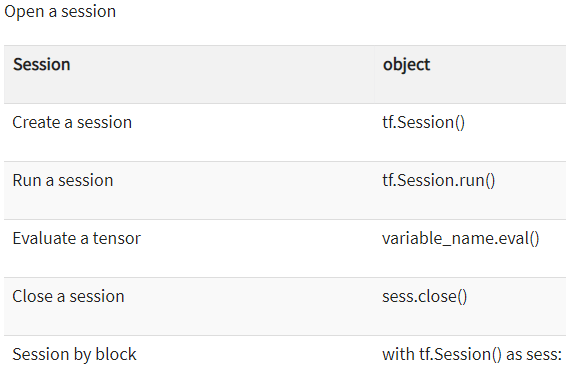












## Why is TensorFlow popular?:

TensorFlow is the best library of all because it is built to be accessible for everyone. Tensorflow library incorporates different API to built at scale deep learning architecture like CNN or RNN. TensorFlow is based on graph computation; it allows the developer to visualize the construction of the neural network with Tensorboad. This tool is helpful to debug the program. Finally, Tensorflow is built to be deployed at scale. It runs on CPU and GPU.

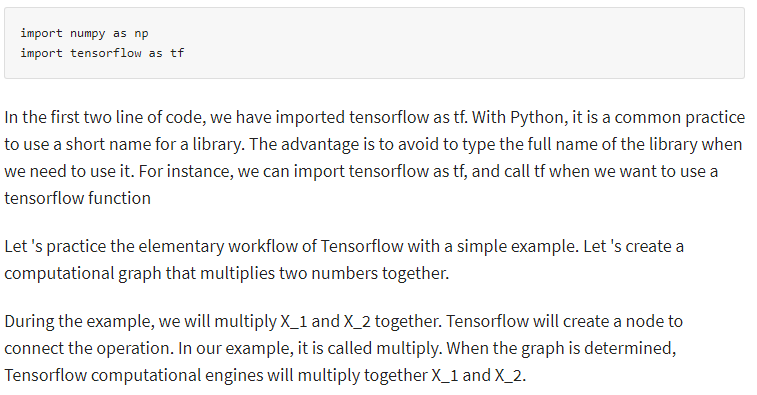
Tensorflow attracts the largest popularity on GitHub compare to the other deep learning framework.

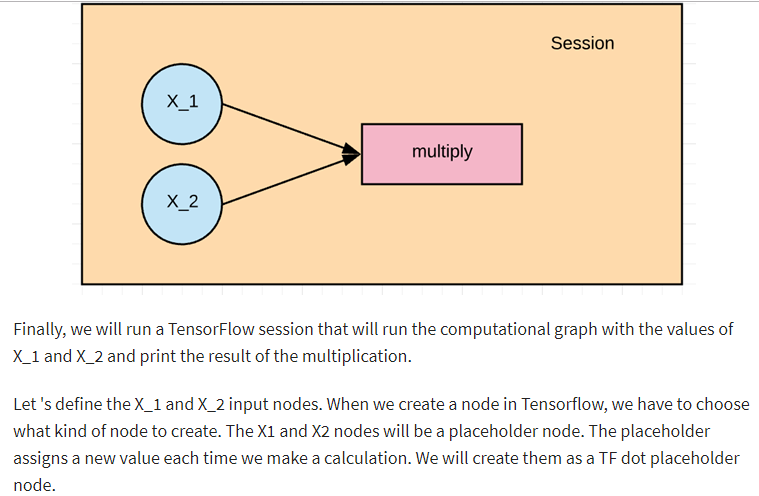
## List of Prominent Algorithms supported by TensorFlow:

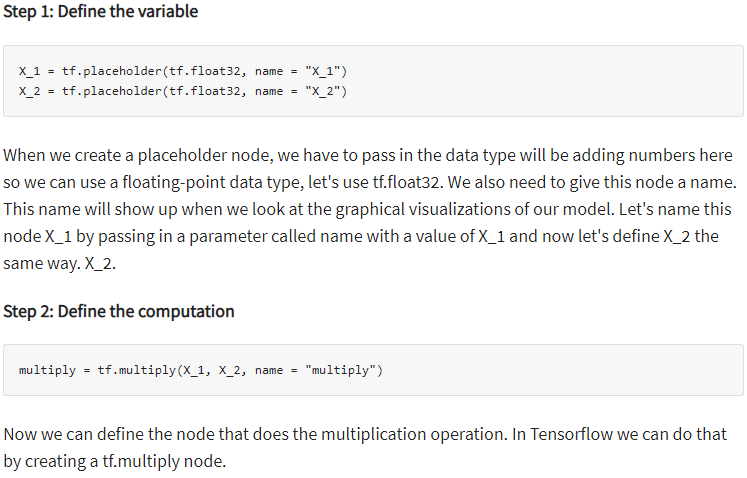
Currently, TensorFlow 1.10 has a built-in API for:

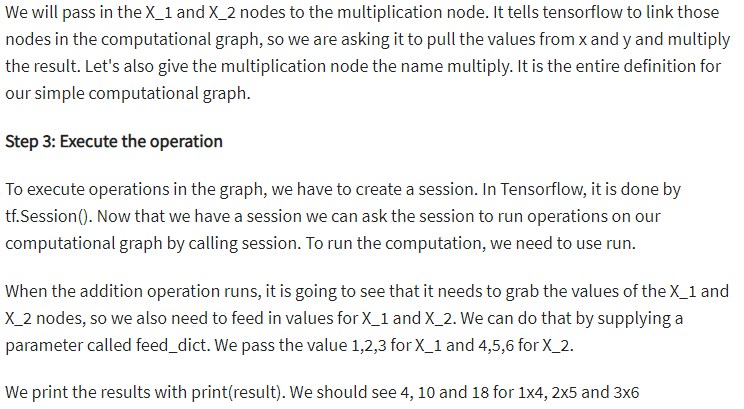
* Linear regression: tf.estimator.LinearRegressor
* Classification:tf.estimator.LinearClassifier
* Deep learning classification: tf.estimator.DNNClassifier
* Deep learning wipe and deep: tf.estimator.DNNLinearCombinedClassifier
* Booster tree regression: tf.estimator.BoostedTreesRegressor
* Boosted tree classification: tf.estimator.BoostedTreesClassifier

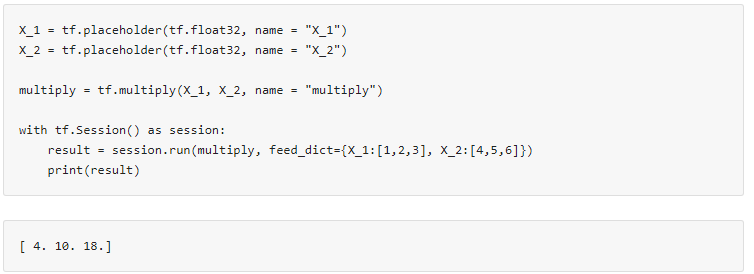
## Simple TensorFlow Example:

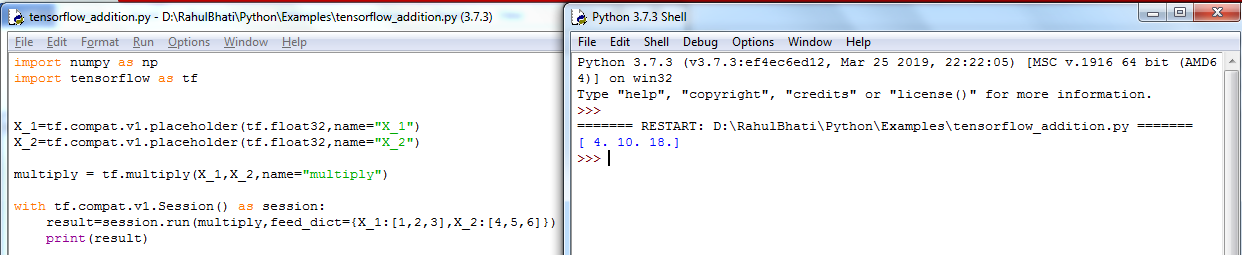












## Options to Load Data into TensorFlow:

The first step before training a machine learning algorithm is to load the data. There is two commons way to load data:

1. Load data into memory: It is the simplest method. You load all your data into memory as a single array. You can write a Python code. This lines of code are unrelated to Tensorflow.

2. Tensorflow data pipeline. Tensorflow has built-in API that helps you to load the data, perform the operation and feed the machine learning algorithm easily. This method works very well especially when you have a large dataset. For instance, image records are known to be enormous and do not fit into memory. The data pipeline manages the memory by itself.

What solution to use?

### Load data in memory:

If your dataset is not too big, i.e., less than 10 gigabytes, you can use the first method. The data can fit into the memory. You can use a famous library called Pandas to import CSV files. You will learn more about pandas in the next tutorial.

### Load data with Tensorflow pipeline:

The second method works best if you have a large dataset. For instance, if you have a dataset of 50 gigabytes, and your computer has only 16 gigabytes of memory then the machine will crash.

In this situation, you need to build a Tensorflow pipeline. The pipeline will load the data in batch, or small chunk. Each batch will be pushed to the pipeline and be ready for the training. Building a pipeline is an excellent solution because it allows you to use parallel computing. It means Tensorflow will train the model across multiple CPUs. It fosters the computation and permits for training powerful neural network.

You will see in the next tutorials on how to build a significant pipeline to feed your neural network.

In a nutshell, if you have a small dataset, you can load the data in memory with Pandas library.

If you have a large dataset and you want to make use of multiple CPUs, then you will be more comfortable to work with Tensorflow pipeline.

### Create Tensorflow pipeline:

In the example before, we manually add three values for X\_1 and X\_2. Now we will see how to load data to Tensorflow.



