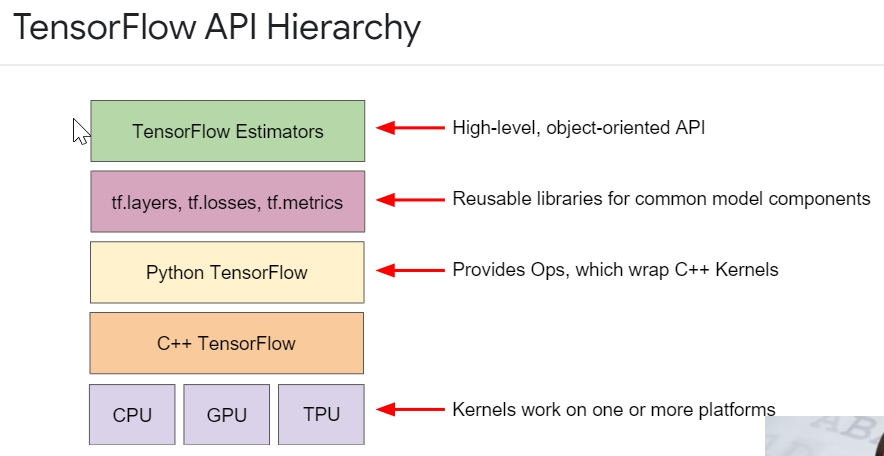
# Tensor flow:



TensorFlow is a computational framework for building machine learning models. TensorFlow provides a variety of different toolkits that allow you to construct models at your preferred level of abstraction. You can use lower-level APIs to build models by defining a series of mathematical operations. Alternatively, you can use higher-level APIs (like tf.estimator) to specify predefined architectures, such as linear regressors or neural networks.

Many of the coding exercises contain the following hyperparameters:

* **steps**, which is the total number of training iterations. One step calculates the loss from *one batch* and uses that value to modify the model's weights *once*.
* **batch size**, which is the number of training examples (chosen at random) for a single step. For example, the batch size for SGD is 1.

The following formula applies:

totalnumberoftrainingexamples=batchsize∗steps

## **A convenience variable in Machine Learning Crash Course exercises**

The following convenience variable appears in several exercises:

* **periods**, which controls the granularity of reporting. For example, if periods is set to 7 and steps is set to 70, then the exercise will output the loss value every 10 steps (or 7 times). Unlike hyperparameters, we don't expect you to modify the value of periods. Note that modifying periods does not alter what your model learns.

The following formula applies:

numberoftrainingexamplesineachperiod=(batchsize∗steps)/periods.

## **Tensor**

#TensorFlow

The primary data structure in TensorFlow programs. Tensors are N-dimensional (where N could be very large) data structures, most commonly scalars, vectors, or matrices. The elements of a Tensor can hold integer, floating-point, or string values.

**What is TensorFlow:** TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

**Feature:** The *input(s)* to our model. In this case, a single value — marketing Budget.

**Labels:** The *output* our model predicts. In this case, a single value — the number of new subscribers gained.

**Example:** A pair of inputs/outputs used during training. In our case a pair of values from mar\_budget and New\_subs at a specific index, such as (80,200).

**Model:**A mathematical representation of a real-world process. In machine Learning a model is an artifact or entity which is created by using a class of algorithm and training it with Features and labels.

**Principle assumption of Linear Regression:**There must be a linear relationship between labels and Coefficients of the equation of line fitted.

* input\_shape=[1]: This specifies that the input to this layer is a single value. That is, the shape is a one-dimensional array with one member. Since this is the first (and only) layer, the input shape is the input shape of the entire model. The single value is a floating-point number, representing marketing\_budget.
* units=1: This specifies the number of neurons in the layer. The number of neurons defines how many internal variables the layer has to try to learn how to solve the problem. Since this is the final layer, it is also the size of the model’s output — a single float value representing new subscribers gained. (In a multi-layered network, the size and shape of the layer would need to match the `input\_shape` of the next layer.)
* Loss function: A way of measuring how far off predictions are from the desired outcome. (The measured difference is called the “loss”.)
* Optimizer function: A way of adjusting internal values to reduce the loss.

1. The loss function ([mean squared error](https://en.wikipedia.org/wiki/Mean_squared_error)) and the optimizer ([Adam](https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/)) used here are standard for simple models like this one, but many others are available. It is not important to know how these specific functions work at this point.
2. One part of the Optimizer you may need to think about when building your models is the learning rate ( 0.1 in the code above). This is the step size taken when adjusting values in the model. If the value is too small, it will take too many iterations to train the model. Too large, and accuracy goes down. Finding a good value often involves some trial and error, but the range is usually within 0.001 (default), and 0.1.

The **epochs** argument specifies how many times this cycle should be run, and the **verbose** argument controls how much output the method produces.

During training, the model takes in marketing budget values, performs a calculation using the current internal variables (called “weights”) and outputs values which are meant to be the `New subs Gained`. Since the weights are initially set randomly, the output will not be close to the correct value. The difference between the actual output and the desired output is calculated using the loss function, and the optimizer function directs how the weights should be adjusted.

This cycle of calculating, comparing, adjusting is controlled by the `fit` method. The first argument is the inputs, the second argument is the desired outputs.

**Verifying the Model accuracy using Performance Metric**

Let’s check the goodness of fit for the model using r2\_score(r-squared value)  
R² is a statistic that will give some information about the goodness of fit of a model. In regression, the R² coefficient of determination is a statistical measure of how well the regression predictions approximate the real data points.

**An** R2**of 1 indicates that the regression predictions perfectly fit the data.**