* **import tensorflow as tf**
* **import numpy as np**
* **import matplotlib.pyplot as plt**
* **x = np.arange(-7.3,20,3)** # make list from -7.3 to 20 step by 3
* y = x + 10
* **x = tf.constant(x)** # create a TensorFlow objects (matrix) that is not changeable..
* **x = tf.Variable(x)** # create a TensorFlow objects (matrix) that can be changed..
* **x.ndim** # to show number of dimensions in the TF matrix.
* **tf.random.set\_seed(42)**  # seed to make random numbers..
* **tf.squeeze(x)** # Removes dimensions of size 1 from the shape of a tensor.
* **tf.expand\_dims(x, axis= -1)**  # used to insert an addition dimension in input Tensor.
* **tf.one\_hot(x, depth=4)** # one hot encoding (new feature for non 0/1 data).
* **tf.reduce\_max(x).numpy()** #  find max/min/sum of elements across dimensions of a tensor.
* **tf.argmax(x)** # Returns index of the largest/smallest’argmin’ value in tensor
* **model = tf.keras.Sequential([tf.keras.layers.Dense(10, input\_shape=[1]),tf.keras.layers.Dense(2)])**

**or**

* **model = tf.keras.Sequential()** # create model of type sequential..
* **model.add(tf.keras.layers.Dense(1, name='input\_layer', input\_shape=[1]))** #add layer to model
* **model.add(tf.keras.layers.Dense(50, name='input\_layer', input\_shape=[1]))**
* **model.add(tf.keras.layers.Dense(1, name='output\_layer'))**
* **model.compile(loss = tf.keras.losses.mae, optimizer= tf.keras.optimizers.SGD(), metrics = ["mae"])**

**or**

* **model.compile(loss = tf.keras.losses.mae, optimizer= tf.keras.optimizers.Adam(lr=0.01), metrics = ["mae"])**
* **model.fit(x,y,epochs=100)** # set training data & epochs..
* **model.predict([20])**  # model output (predict a value by model)
* **plt.scatter(x,y)** # draw a graph in pyplotlib library
* x\_train = x[:40]
* y\_train = y[:40]
* x\_test = x[40:]
* y\_test = y[40:]
* **plt.figure(figsize=(10,7))** # graph size..
* **plt.scatter(x\_train, y\_train , c='b', label="training data")** # draw train data on graph with blue color.
* **plt.scatter(x\_test, y\_test , c='r', label="testing data")** # draw test data on graph with red color.
* **plt.legend()** # show the drown graph..
* y\_preds = model.predict(x\_test)
* tf.keras.utils.plot\_model(model, show\_shapes=True) # draw an illustration of model
* model.summary() # print model details.