In [1]:
```
import numpy as np
import tensorflow as tf
```

In [2]:
```
#None defines shape if [2,None] and [2,2] than you fix both we use None for dynamic
x=tf.placeholder('float',None)
y=x+10                             #here you can also use tf.add(x)+10 also
```

In [5]:
```
print(x)
print(y)
```

```
Tensor("Placeholder:0", dtype=float32)
Tensor("add:0", dtype=float32)
```

In [6]:
```
with tf.Session() as sess:    #fill methods is known as feed
    re=sess.run(y,feed_dict={x:5})
    print(re)
sess.close()
```

```
15.0
```

In [7]:
```
with tf.Session() as sess:      #fill as many values you want
         re=sess.run(y,feed_dict={x:[5,4,4,4]})
         print(re)
sess.close()
```

```
[15. 14. 14. 14.]
```

In [8]:
```
#variable ki value we cannot fill by Session
x=tf.placeholder(tf.float32,[None,1])
w=tf.Variable(tf.zeros([1,1]))
b=tf.Variable(tf.zeros([1]))
product=tf.matmul(x,w)
Yp=tf.add(product,b)
Ya=tf.placeholder(tf.float32,[None,1])              #question as x answer always 1
cost_f=tf.reduce_mean(tf.square(Ya-Yp))             #cost function reduce by gradient descend
init=tf.global_variables_initializer()              #this method initialize the variable
training_step=tf.train.GradientDescentOptimizer(.001).minimize(cost_f)
```

In [16]:
```
with tf.Session() as sess:
    sess.run(init)
    steps=100
    for i in range(steps):
        Xs=np.array([[i]])
        Ys=np.array([[2*i]])
        feed={x:Xs,Ya:Ys}
        sess.run(training_step,feed_dict=feed)
        print("After {} iteration :".format(i))
        print("w : %f"% sess.run(w))
        print("b :%f"% sess.run(b))
        print("cost :%f"% sess.run(cost_f,feed_dict=feed))
```

```
After 0 iteration :
w : 0.000000
b :0.000000
cost :0.000000
After 1 iteration :
w : 0.004000
b :0.004000
cost :3.968064
After 2 iteration :
w : 0.019952
b :0.011976
cost :15.587652
After 3 iteration :
w : 0.055521
b :0.023832
cost :33.751507
After 4 iteration :
w : 0.117554
b :0.039341
```

In [ ]: