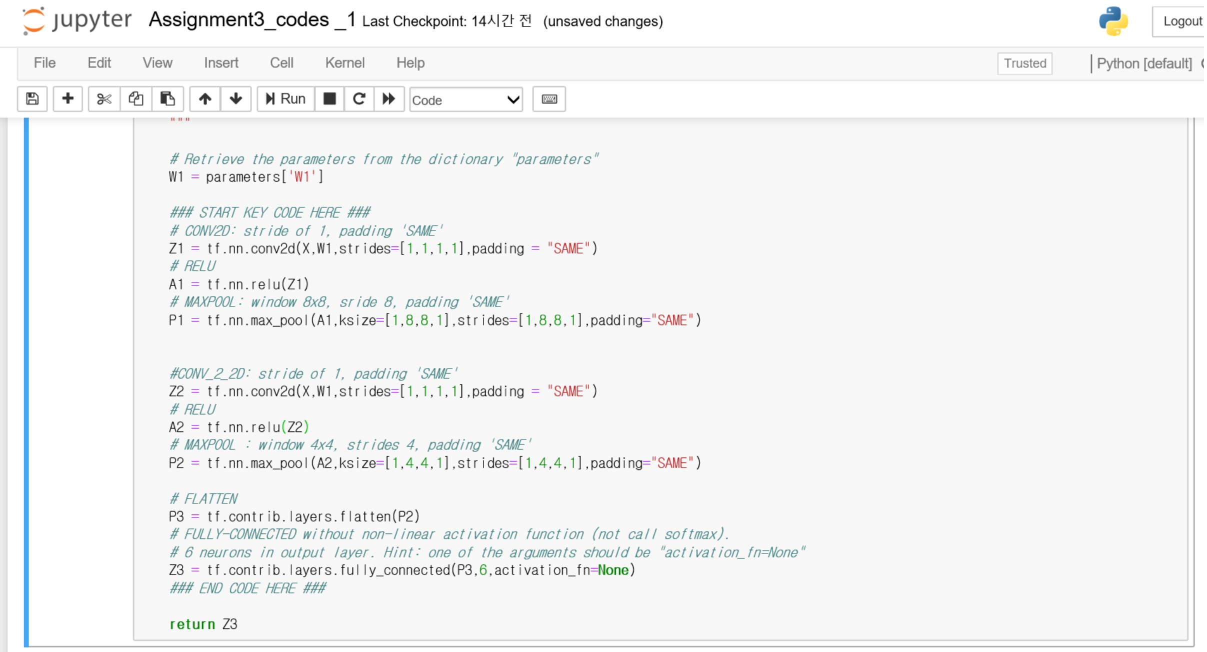
1.

1) Briefly describe how you added the layer.




```
    # Retrieve the parameters from the dictionary "parameters"
    W1 = parameters['W1']

    ### START KEY CODE HERE ###
    # CONV2D: stride of 1, padding 'SAME'
    Z1 = tf.nn.conv2d(X,W1,strides=[1,1,1,1],padding = "SAME")
    # RELU
    A1 = tf.nn.relu(Z1)
    # MAXPOOL: window 8x8, sride 8, padding 'SAME'
    P1 = tf.nn.max_pool(A1,ksize=[1,8,8,1],strides=[1,8,8,1],padding="SAME")


    #CONV_2_2D: stride of 1, padding 'SAME'
    Z2 = tf.nn.conv2d(X,W1,strides=[1,1,1,1],padding = "SAME")
    # RELU
    A2 = tf.nn.relu(Z2)
    # MAXPOOL : window 4x4, strides 4, padding 'SAME'
    P2 = tf.nn.max_pool(A2,ksize=[1,4,4,1],strides=[1,4,4,1],padding="SAME")

    # FLATTEN
    P3 = tf.contrib.layers.flatten(P2)
    # FULLY-CONNECTED without non-linear activation function (not call softmax).
    # 6 neurons in output layer. Hint: one of the arguments should be "activation_fn=None"
    Z3 = tf.contrib.layers.fully_connected(P3,6,activation_fn=None)
    ### END CODE HERE ###

    return Z3
```

We added layer2 between layer1 and flatten. The followings are what we added.

#CONV_2_2D: stride of 1, padding 'SAME'

Z2 = tf.nn.conv2d(X,W1,strides=[1,1,1,1],padding = "SAME")

RELU

A2 = tf.nn.relu(Z2)

MAXPOOL : window 4x4, strides 4, padding 'SAME'

P2 = tf.nn.max_pool(A2,ksize=[1,4,4,1],strides=[1,4,4,1],padding="SAME")

As we added P2, We changed original P2 to P3.

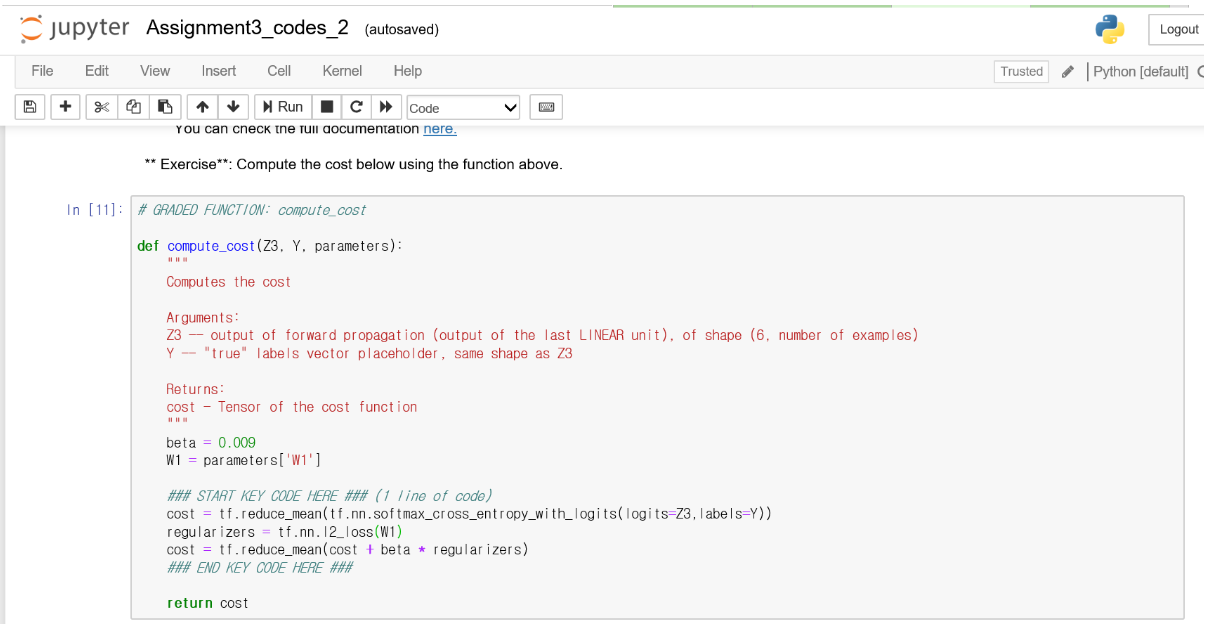
2) Report your train and test accuracy (%) by running the following commend. Write the average accuracy of the total of 5 runs (e.g., 80% = (79+80+81+78+82)/5). Each run has 150 epochs (iteration).

Train : 0.166667

Test : 0.166667

2.

1) Briefly describe how you implemented the regularization term.




```
# GRADED FUNCTION: compute_cost

def compute_cost(Z3, Y, parameters):
    """
    Computes the cost

    Arguments:
    Z3 -- output of forward propagation (output of the last LINEAR unit), of shape (6, number of examples)
    Y -- "true" labels vector placeholder, same shape as Z3

    Returns:
    cost - Tensor of the cost function
    """
    beta = 0.009
    W1 = parameters['W1']

    ### START KEY CODE HERE ### (1 line of code)
    cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=Z3,labels=Y))
    regularizers = tf.nn.l2_loss(W1)
    cost = tf.reduce_mean(cost + beta * regularizers)
    ### END KEY CODE HERE ###

    return cost
```

We wanted to regularize W1, so we added the following codes

Regularizes = tf.nn.l2_loss(W1)

Cost=tf . reduce _ mean(cost + beta = regularizers)

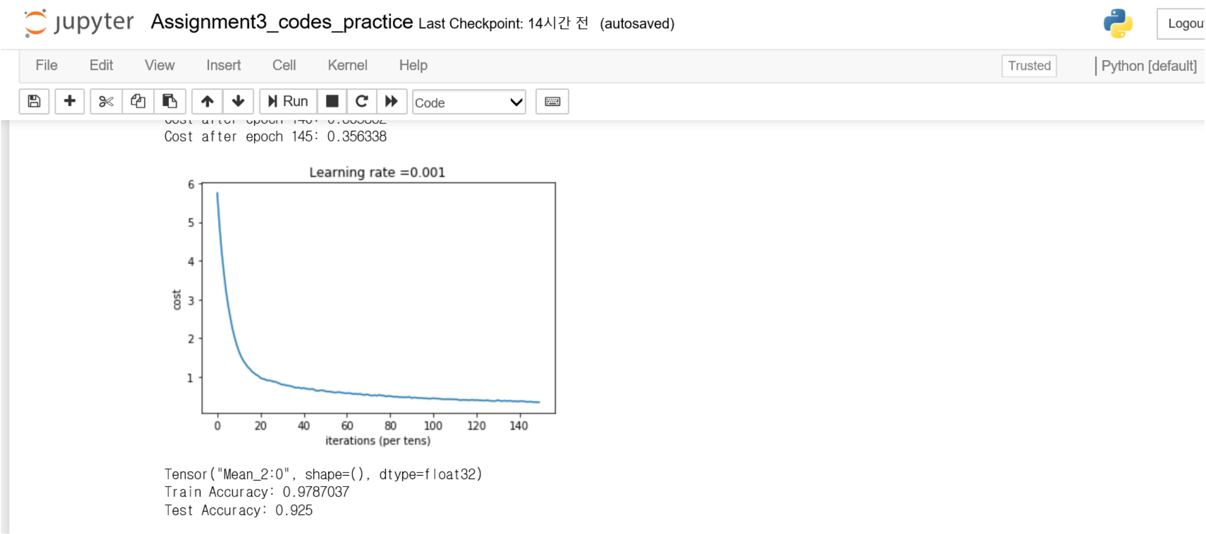
2) Report your train and test accuracy (%) by running the following commend. Write the average accuracy of the total of 5 runs. Each run has 150 epochs (iteration).

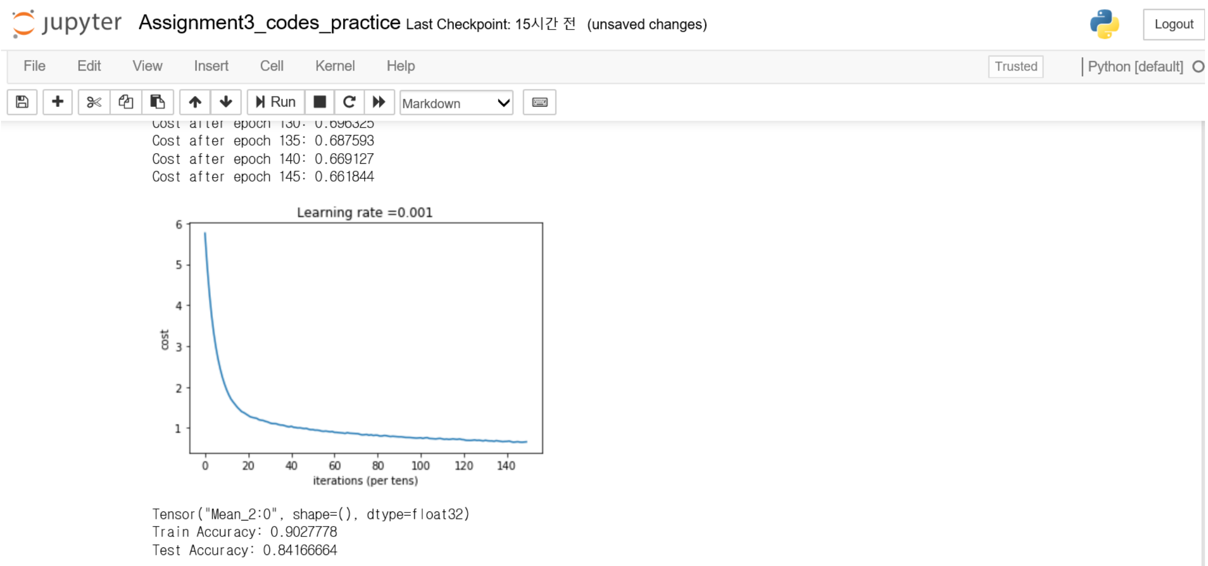
Train : 0.982407

Test : 0.808333

3.

1) Describe how you improved its “test” accuracy (e.g., tuning the hyperparameters, changing the structure of layers).





2) Report your test accuracy (%) by running the following commend. Write the average accuracy of the total of 5 runs. You may choose the number of epochs in your discretion, but it should not exceed 1000 epochs per run. ### START KEY CODE HERE ### (1 line of code)

Train : 1.0

Test : 0.775

Train : 0.910185

Test : 0.858333