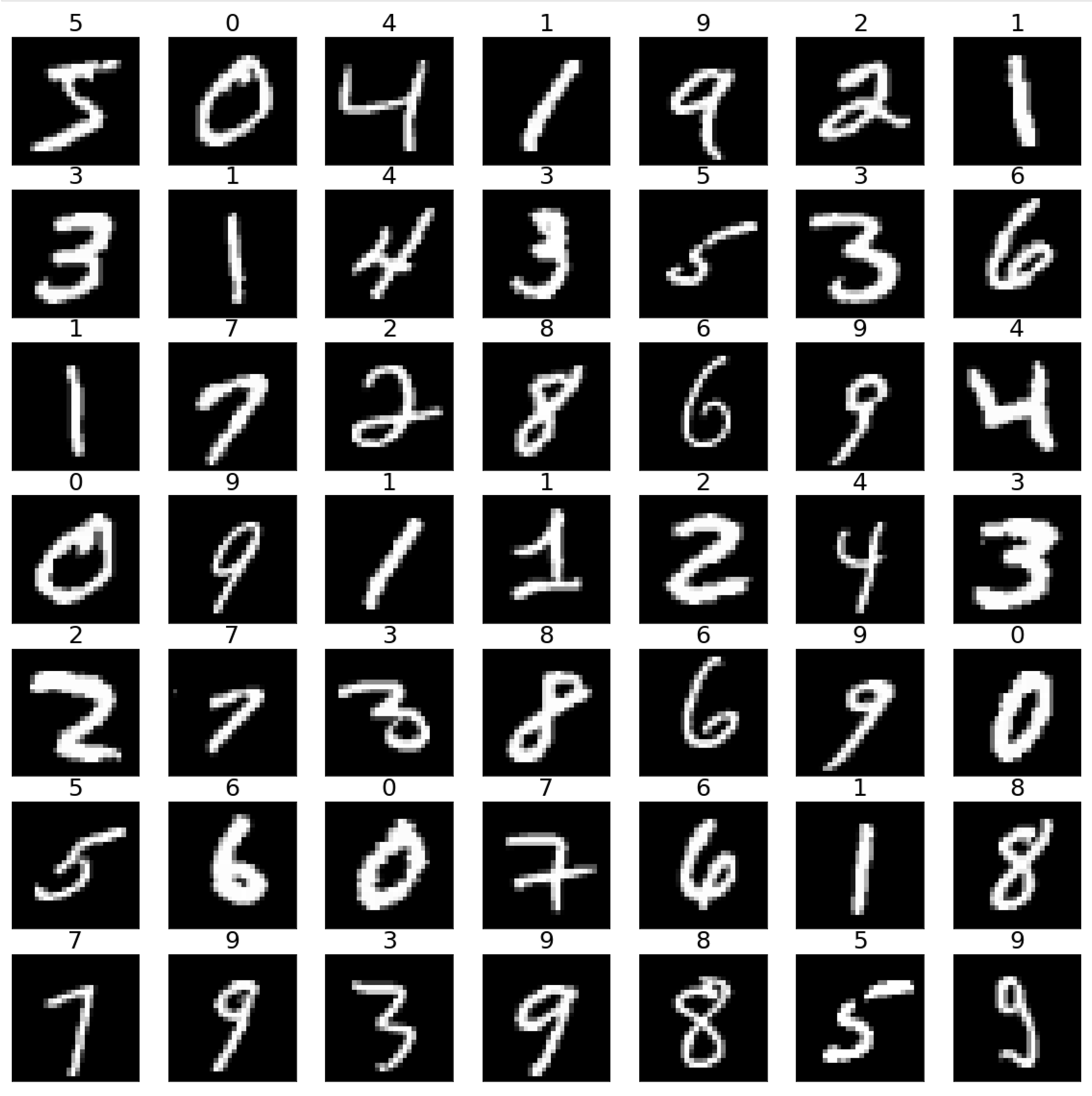
In [66]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [67]:
```python
from tensorflow.keras.datasets import mnist
```

In [68]:
```python
(x_train,y_train),(x_test,y_test) = mnist.load_data()
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

In [69]:
```python
plt.figure(figsize=(20,20))
for i in range(49):
    plt.subplot(7,7,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_train[i],'gray')
    plt.title(y_train[i],fontdict={'size': 25})
plt.show()
```



In [70]:
```python
x_train = x_train[ :10000,:,:]
x_test = x_test[ :5000,:,:]
y_train = y_train[ :10000]
y_test = y_test[ :5000]
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(10000, 28, 28)
(5000, 28, 28)
(10000,)
(5000,)
```

In [71]:
```python
x_train[0]
```

Out[71]:
```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
        253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
        205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
         90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
        190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
        253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
        241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
        148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
        253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
        253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
        195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
         11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0]], dtype=uint8)
```

1. Normalization of Images

In [72]:
```python
x_train = x_train/255
x_test = x_test/255

print(x_train.shape)
print(x_test.shape)
```

```
(10000, 28, 28)
(5000, 28, 28)
```

2. Building CNN Architecture

In [73]:
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropout
```

In [74]:
```python
s1 = Sequential()
s1.add(Conv2D(48,(3,3),activation='relu',input_shape=(28,28,1) ))
s1.add(MaxPooling2D(pool_size=(2,2)))

s1.add(Conv2D(48,(3,3),activation='relu'))
s1.add(MaxPooling2D(pool_size=(2,2)))

s1.add(Conv2D(48,(3,3),activation='relu'))
s1.add(MaxPooling2D(pool_size=(2,2)))

s1.add(Flatten())
s1.add(Dense(24,activation='relu'))
s1.add(Dense(12,activation='relu'))
s1.add(Dense(10,activation='softmax'))

s1.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['acc'])
```

3. Executing Model For Appropriate Number of Epochs

In [75]:
```python
t1 = s1.fit(x_train,y_train,epochs=30,validation_data=(x_test,y_test))
```

```
Epoch 1/30
313/313 [==============================] - 4s 12ms/step - loss: 0.8441 - acc: 0.7340 - val_loss: 0.3735 - val_acc: 0.8800
Epoch 2/30
313/313 [==============================] - 4s 12ms/step - loss: 0.2698 - acc: 0.9229 - val_loss: 0.2654 - val_acc: 0.9184
Epoch 3/30
313/313 [==============================] - 4s 11ms/step - loss: 0.1856 - acc: 0.9466 - val_loss: 0.2146 - val_acc: 0.9308
Epoch 4/30
313/313 [==============================] - 4s 12ms/step - loss: 0.1410 - acc: 0.9591 - val_loss: 0.2572 - val_acc: 0.9250
Epoch 5/30
313/313 [==============================] - 4s 11ms/step - loss: 0.1071 - acc: 0.9680 - val_loss: 0.1719 - val_acc: 0.9448
Epoch 6/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0959 - acc: 0.9714 - val_loss: 0.1908 - val_acc: 0.9440
Epoch 7/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0787 - acc: 0.9767 - val_loss: 0.1730 - val_acc: 0.9480
Epoch 8/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0762 - acc: 0.9784 - val_loss: 0.1454 - val_acc: 0.9546
Epoch 9/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0595 - acc: 0.9821 - val_loss: 0.2121 - val_acc: 0.9408
Epoch 10/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0559 - acc: 0.9839 - val_loss: 0.2059 - val_acc: 0.9460
Epoch 11/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0510 - acc: 0.9837 - val_loss: 0.1659 - val_acc: 0.9558
Epoch 12/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0362 - acc: 0.9890 - val_loss: 0.1510 - val_acc: 0.9578
Epoch 13/30
313/313 [==============================] - 4s 11ms/step - loss: 0.0337 - acc: 0.9894 - val_loss: 0.1751 - val_acc: 0.9520
Epoch 14/30
313/313 [==============================] - 4s 11ms/step - loss: 0.0369 - acc: 0.9877 - val_loss: 0.1908 - val_acc: 0.9470
Epoch 15/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0289 - acc: 0.9906 - val_loss: 0.1518 - val_acc: 0.9600
Epoch 16/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0270 - acc: 0.9914 - val_loss: 0.1576 - val_acc: 0.9626
Epoch 17/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0271 - acc: 0.9910 - val_loss: 0.2330 - val_acc: 0.9456
Epoch 18/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0178 - acc: 0.9943 - val_loss: 0.1762 - val_acc: 0.9608
Epoch 19/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0161 - acc: 0.9947 - val_loss: 0.1694 - val_acc: 0.9610
Epoch 20/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0213 - acc: 0.9929 - val_loss: 0.2354 - val_acc: 0.9492
Epoch 21/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0293 - acc: 0.9900 - val_loss: 0.1885 - val_acc: 0.9522
Epoch 22/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0078 - acc: 0.9980 - val_loss: 0.1857 - val_acc: 0.9608
Epoch 23/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0169 - acc: 0.9935 - val_loss: 0.1632 - val_acc: 0.9642
Epoch 24/30
313/313 [==============================] - 4s 11ms/step - loss: 0.0216 - acc: 0.9928 - val_loss: 0.1566 - val_acc: 0.9672
Epoch 25/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0064 - acc: 0.9982 - val_loss: 0.1922 - val_acc: 0.9618
Epoch 26/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0095 - acc: 0.9965 - val_loss: 0.2319 - val_acc: 0.9510
Epoch 27/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0118 - acc: 0.9962 - val_loss: 0.1805 - val_acc: 0.9630
Epoch 28/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0192 - acc: 0.9936 - val_loss: 0.1754 - val_acc: 0.9642
Epoch 29/30
313/313 [==============================] - 4s 11ms/step - loss: 0.0065 - acc: 0.9979 - val_loss: 0.1712 - val_acc: 0.9672
Epoch 30/30
313/313 [==============================] - 4s 12ms/step - loss: 0.0019 - acc: 0.9994 - val_loss: 0.2168 - val_acc: 0.9602
```

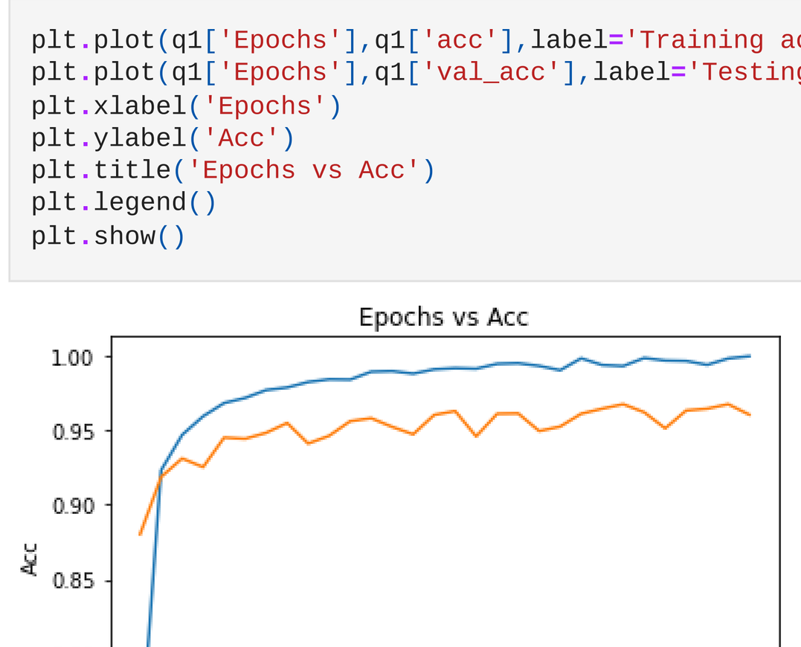
In [76]:
```python
q1 = pd.DataFrame(t1.history)
q1['Epochs'] = t1.epoch
q1.tail()
```

Out[76]:

	loss	acc	val_loss	val_acc	Epochs
25	0.009501	0.9965	0.231871	0.9510	25
26	0.011834	0.9962	0.180473	0.9630	26
27	0.019162	0.9936	0.175431	0.9642	27
28	0.006474	0.9979	0.171185	0.9672	28
29	0.001899	0.9994	0.216790	0.9602	29

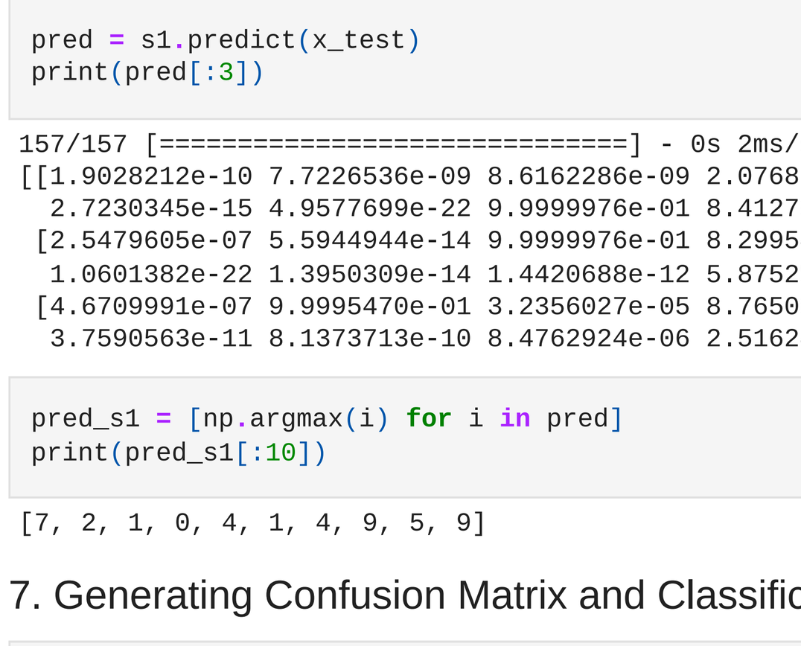
4. Depicting loss Vs val_loss

In [84]:
```python
plt.plot(q1['Epochs'],q1['loss'],label='Training loss')
plt.plot(q1['Epochs'],q1['val_loss'],label='Testing loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Epochs Vs Loss')
plt.legend()
plt.show()
```



5. Depicting acc Vs val_acc

In [85]:
```python
plt.plot(q1['Epochs'],q1['acc'],label='Training acc')
plt.plot(q1['Epochs'],q1['val_acc'],label='Testing acc')
plt.xlabel('Epochs')
plt.ylabel('Acc')
plt.title('Epochs vs Acc')
plt.legend()
plt.show()
```



6. Generating Predictions on Test Data

In [86]:
```python
pred = s1.predict(x_test)
print(pred[:3])
```

```
157/157 [==============================] - 0s 2ms/step
[[1.9028212e-10 7.7226536e-09 8.6162286e-09 2.0768140e-07 4.8074524e-15
  2.7230345e-15 4.9577699e-22 9.9999976e-01 8.4127915e-14 1.7721449e-12]
 [2.5479605e-07 5.5944944e-14 9.9999976e-01 8.2995416e-13 4.6933997e-12
  1.0601382e-22 1.3950309e-14 1.4420688e-12 5.8752551e-12 4.8577564e-13]
 [4.6709991e-07 9.9995470e-01 3.2356027e-05 8.7650838e-09 3.6273659e-06
  3.7590563e-11 8.1373713e-10 8.4762924e-06 2.5162467e-07 5.2624419e-08]]
```

In [87]:
```python
pred_s1 = [np.argmax(i) for i in pred]
print(pred_s1[:10])
```

```
[7, 2, 1, 0, 4, 1, 4, 9, 5, 9]
```

7. Generating Confusion Matrix and Classification Model

In [88]:
```python
from sklearn.metrics import confusion_matrix,classification_report
```

In [89]:
```python
class_m = confusion_matrix(y_test,pred_s1)
print(class_m)
print(classification_report(y_test,pred_s1))
```

```
[[456   0   0   0   0   1   1   0   1   1]
 [  1 558   1   4   0   0   1   5   1   0]
 [  4   0 511   1   4   0   0   9   1   0]
 [  1   0   8 471   0   5   0  14   0   1]
 [  0   0   1   0 494   0   0   1   0   4]
 [  3   0   0   8   0 434   0   1   8   2]
 [ 29   3   0   0   8   2 415   0   5   0]
 [  1   3  15   1   0   0   0 487   1   4]
 [  5   0   6   0   1   0   0   3 469   5]
 [  4   0   1   0   1   0   0   1   7 506]]
              precision    recall  f1-score   support

           0       0.90      0.99      0.95       460
           1       0.99      0.98      0.98       571
           2       0.94      0.96      0.95       530
           3       0.97      0.94      0.96       500
           4       0.97      0.99      0.98       500
           5       0.98      0.95      0.97       456
           6       1.00      0.90      0.94       462
           7       0.93      0.95      0.94       512
           8       0.95      0.96      0.96       489
           9       0.97      0.97      0.97       520

    accuracy                           0.96      5000
   macro avg       0.96      0.96      0.96      5000
weighted avg       0.96      0.96      0.96      5000
```