In [2]:
```
import numpy as np
import tensorflow as ts
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Flatten,Conv2D,MaxPooling2D
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
```

In [3]:
```
(x_train,y_train),(x_test,y_test) = mnist.load_data()
```

In [4]:
```
print(x_train.shape)
print(y_train.shape)
```

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

In [12]:
```
x_train[0].min(),x_train[0].max()
```

Out[12]:
```
(0, 255)
```

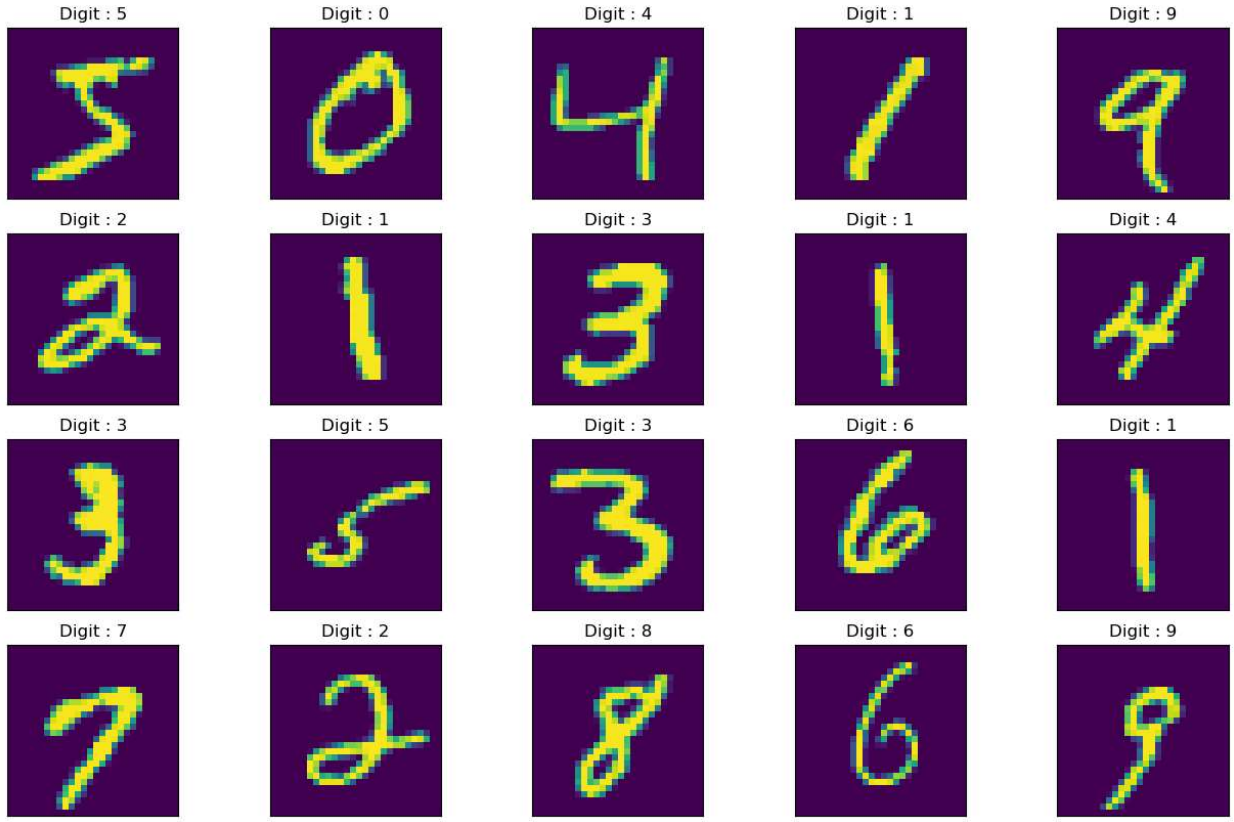
In [13]:
```
x_train,x_test = x_train/255.0,x_test/255.0
```

In [21]:
```
def plot_dig(img,dig,plt,i):
    plt.subplot(4,5,i+1)
    plt.imshow(img)
    plt.title(f"Digit : {dig}")
    plt.xticks([])
    plt.yticks([])

plt.figure(figsize=(16,10))

for i in range(20):
    plot_dig(x_train[i],y_train[i],plt,i)

plt.show()
```



In [22]:
```
x_train = x_train.reshape((x_train.shape+(1,)))
x_test = x_test.reshape((x_test.shape+(1,)))
```

In [24]:
```
model = Sequential([
    Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)),
    MaxPooling2D((2,2)),
    Flatten(),
    Dense(100,activation='relu'),
    Dense(10,activation='softmax')
])
```

```
C:\ProgramData\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.p
y:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Whe
n using Sequential models, prefer using an `Input(shape)` object as the first layer i
n the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [25]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
flatten (Flatten)	(None, 5408)	0
dense (Dense)	(None, 100)	540,900
dense_1 (Dense)	(None, 10)	1,010

Total params: 542,230 (2.07 MB)

Trainable params: 542,230 (2.07 MB)

Non-trainable params: 0 (0.00 B)

In [26]:
```
model.compile(optimizer=SGD(learning_rate=0.01)
              ,loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

In [27]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
flatten (Flatten)	(None, 5408)	0
dense (Dense)	(None, 100)	540,900
dense_1 (Dense)	(None, 10)	1,010

Total params: 542,230 (2.07 MB)

Trainable params: 542,230 (2.07 MB)

Non-trainable params: 0 (0.00 B)

In [28]:
```
model.fit(x_train,y_train,epochs=7)
```

```
Epoch 1/7
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.7622 - loss: 0.8960
Epoch 2/7
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9278 - loss: 0.2406
Epoch 3/7
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9459 - loss: 0.1811
Epoch 4/7
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9572 - loss: 0.1434
Epoch 5/7
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.9614 - loss: 0.1249
Epoch 6/7
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9683 - loss: 0.1064
Epoch 7/7
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9721 - loss: 0.0939
```

Out[28]:

```
<keras.src.callbacks.history.History at 0x236a7759fd0>
```

In [29]:

```
test_loss,test_accuracy = model.evaluate(x_test,y_test)

print(f"Test Loss: {test_loss}, Test Accuracy: {test_accuracy}")
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9683 - loss: 0.1048
Test Loss: 0.09347008913755417, Test Accuracy: 0.972100019454956
```

In [ ]: