```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
import random

# Load the MNIST dataset
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Dataset info
print("Training data shape:", x_train.shape)
print("Testing data shape:", x_test.shape)

# Display first sample image
plt.matshow(x_train[0])
plt.title("Sample Training Image")
plt.show()

# Normalize dataset (scales values 0–255 to 0–1)
x_train = x_train / 255.0
x_test = x_test / 255.0

# Define the Feedforward Neural Network model
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),  # Input Layer
    keras.layers.Dense(128, activation='relu'),  # Hidden Layer
    keras.layers.Dense(10, activation='softmax') # Output Layer (10 digits)
])

# Show model summary
model.summary()

# Compile model
model.compile(optimizer='sgd',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train model
history = model.fit(x_train, y_train, epochs=10,
                    validation_data=(x_test, y_test))

# Evaluate model
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"\nTest Loss = {test_loss:.3f}")
print(f"Test Accuracy = {test_acc:.3f}")

# Predict a random test image
n = random.randint(0, len(x_test)-1)
plt.imshow(x_test[n], cmap='gray')
plt.title("Actual Digit: {}".format(y_test[n]))
plt.show()
```

```
pred = model.predict(x_test)
print("Predicted Digit =", np.argmax(pred[n]))

# Plot training graphs
plt.figure(figsize=(8,5))
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Test Accuracy')
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Test Loss')
plt.title("Training History")
plt.xlabel("Epoch")
plt.ylabel("Accuracy / Loss")
plt.legend()
plt.show()
```