Ex. No. 5	**RNN**
Date of Exercise	**09/09/2025**

Aim:

To implement a Recurrent Neural Network (RNN) for predicting future values in a time series dataset, such as stock prices.

Description:

RNNs are powerful neural networks for handling sequential data, making them ideal for time series forecasting. We use a dataset containing stock prices and train an RNN model to predict future values based on historical trends.

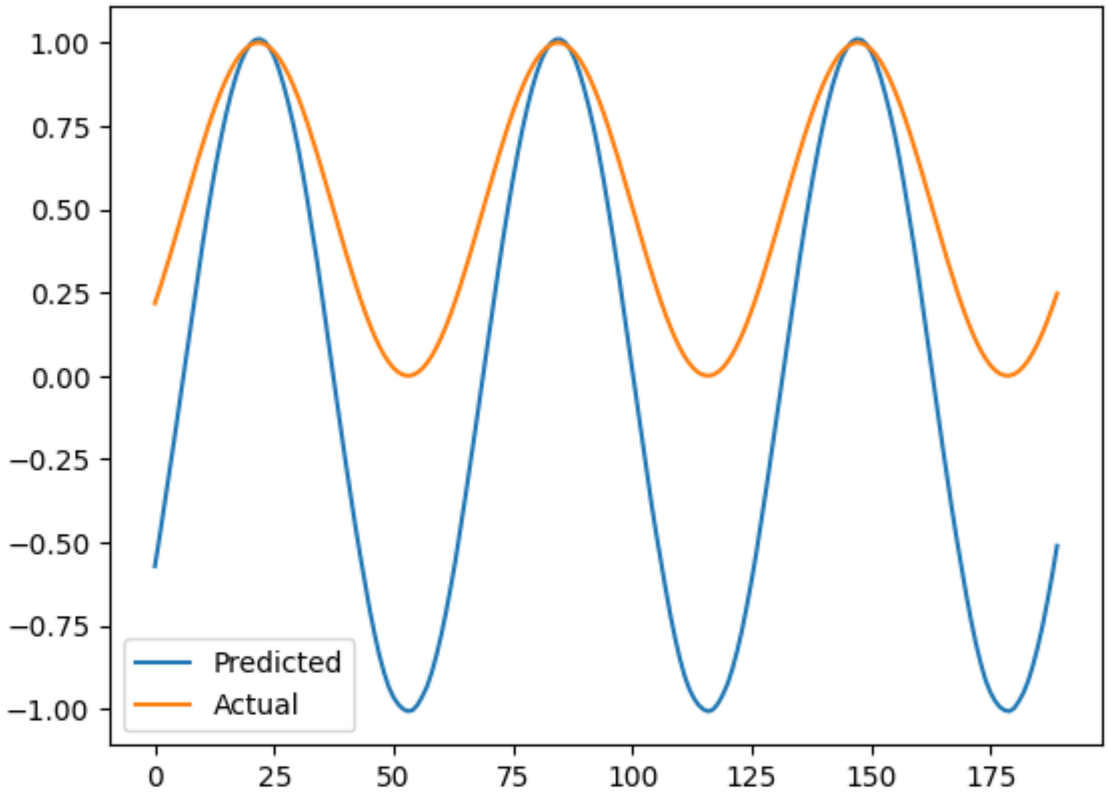
Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.optimizers import Adam
# Generate synthetic time series data
data = np.sin(np.linspace(0, 100, 1000))
data = data.reshape(-1, 1)
# Scaling data
```

```
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data)
# Prepare dataset
def create_sequences(data, seq_length):
X, y = [], []
for i in range(len(data) - seq_length):
X.append(data[i:i+seq_length])
y.append(data[i+seq_length])
return np.array(X), np.array(y)
seq_length = 10
X, y = create_sequences(data_scaled, seq_length)

# Split data
X_train, X_test = X[:800], X[800:]
y_train, y_test = y[:800], y[800:]
# Build RNN model
model = Sequential([
SimpleRNN(50, activation='relu', return_sequences=True, input_shape=(seq_length, 1)),
SimpleRNN(50, activation='relu'),
Dense(1)
])
model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')
model.fit(X_train, y_train, epochs=20, batch_size=16, validation_data=(X_test, y_test))
# Predict
y_pred = model.predict(X_test)
y_pred_inv = scaler.inverse_transform(y_pred)
# Plot results
plt.plot(y_pred_inv, label='Predicted')
plt.legend()
plt.show()
```

Sample Output:



Result

The code for RNN is Done successful and the output is been verified