```
import numpy as np
t=np.linspace(0,150,100)
data=np.sin(t)
```

This cell initializes a NumPy array t representing 100 evenly spaced values between 0 and 150. Then, it calculates the sine of each value in t and stores the results in the data array.

```
window_size = 10
x=[]
y=[]

for i in range(len(data)-window_size):
  x.append(data[i:i+window_size])
  y.append(data[i+window_size])

x=np.array(x)
y=np.array(y)
```

This code prepares the data for time series modeling. It defines a window_size of 10 and then iterates through the data array to create sequences. For each iteration, it extracts a slice of 10 data points as an input sequence (x) and the very next data point as the corresponding target (y). Finally, it converts these lists of sequences and targets into NumPy arrays, which are suitable for training machine learning models.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense

model=Sequential()
model.add(SimpleRNN(16,input_shape=(window_size, 1)))
model.add(Dense(1))
model.compile(optimizer='adam',loss='mse')
model.fit(x.reshape(-1, window_size, 1),y,epochs=10)
```

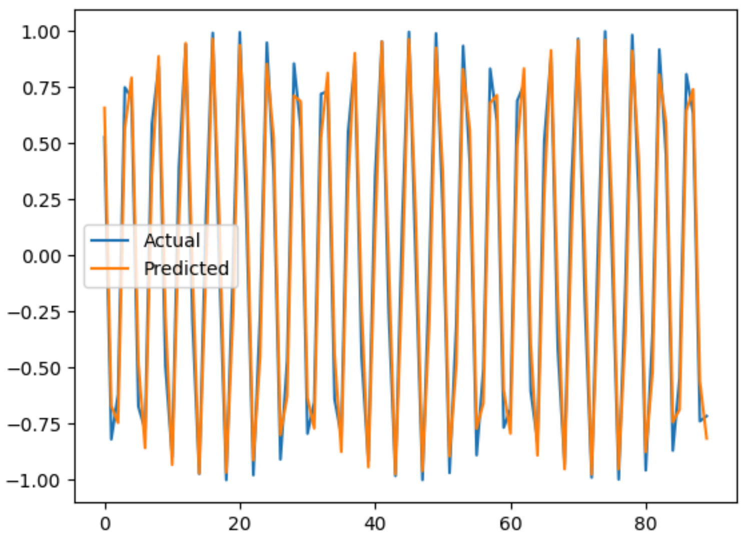
```
Epoch 1/10
/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `input_shape
  super().__init__(**kwargs)
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - loss: 1.1186
Epoch 2/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.9681
Epoch 3/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.7192
Epoch 4/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.5902
Epoch 5/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.4613
Epoch 6/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.3627
Epoch 7/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - loss: 0.2218
Epoch 8/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - loss: 0.1563
Epoch 9/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - loss: 0.0890
Epoch 10/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - loss: 0.0509
<keras.src.callbacks.history.History at 0x7c55845baa80>
```

This code sets up and trains a simple Recurrent Neural Network (RNN) model using Keras. It imports necessary layers, defines a sequential model with one SimpleRNN layer and one Dense output layer, compiles the model with the 'adam' optimizer and 'mse' (mean squared error) loss, and then trains it for 10 epochs using the prepared x and y data. The input_shape is set based on the window_size and indicates that each input sequence has window_size elements with a single feature.

```
import matplotlib.pyplot as plt

predictions = model.predict(x.reshape(-5, window_size, 1))
plt.plot(y, label='Actual')
plt.plot(predictions.flatten(), label='Predicted')
plt.legend()
plt.show()
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 77ms/step
```



This code block is responsible for visualizing the performance of the trained RNN model. It first imports the matplotlib.pyplot library for plotting. Then, it uses the model.predict() method to generate predictions on the input data x, after reshaping x to the format expected by the model. Finally, it creates a plot comparing the actual target values (y) with the model's predictions, adds a legend for clarity, and displays the plot.