Code to implement RNN

Import the required libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

Load the dataset

```
data = pd.read_csv('GOOG.csv')
```

Prepare the data

```
# Extract the 'Open' column
dataset = data['Open'].values.reshape(-1, 1)
# Scale the data between 0 and 1
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
# Create the training and testing datasets
training_data_len = int(len(dataset) * 0.8)
training_data = dataset[:training_data_len]
testing_data = dataset[training_data_len:]
def create_dataset(dataset, time_step=1):
X, Y = [], []
for i in range(len(dataset) - time_step - 1):
X.append(dataset[i:(i+time_step), 0])
Y.append(dataset[i+time_step, 0])
return np.array(X), np.array(Y)
# Create the training and testing datasets with a time step of 60 days
time_step = 60
X_train, Y_train = create_dataset(training_data, time_step)
X_test, Y_test = create_dataset(testing_data, time_step)
```

```
# Reshape the training and testing datasets
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

Create the RNN model

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(
```

Output:

```
Epoch 100/100
33/33 [==============================] - 7s 39ms/step - loss: 0.0013
```