In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:

```
from google.colab import drive
drive.mount('/gdrive', force_remount=True)
```

In [ ]:

```
df = pd.read_csv('/gdrive/My Drive/Keras/gangnamgu.csv', parse_dates =["date"], index_col ="dat
e", dtype={'trade_price_idx_value':'float'})

df.head()
```

In [ ]:

```
df.info()
```

In [ ]:

```
df['trade_price_idx_value'].plot()
```

In [ ]:

```
df
```

In [ ]:

```
# 훈련 데이터: 2017/1/1 까지의 데이터
# 시험 데이터: 그 이후 데이터

split_date = pd.Timestamp('01-01-2017')

train = df.loc[:split_date, ['trade_price_idx_value']]
test = df.loc[split_date:, ['trade_price_idx_value']]

ax = train.plot()
test.plot(ax=ax)
plt.legend(['train', 'test'])
```

In [ ]:

```
# 데이터 스케일링(Scaling)
# MinMaxScaler 클래스를 사용하여 데이터 스케일링
# MinMaxScaler(X)는 데이터의 최대 값이 1, 최소값이 0이 되도록 변환

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler()

# 훈련 데이터의 분포 추정, 훈련 데이터 변환
train_sc = sc.fit_transform(train)
# 시험 데이터 변환
test_sc = sc.transform(test)

train_sc
```

In [ ]:

```
test_sc
```

In [ ]:

```
# 정규화가 완료된 데이터들은 다시 pandas dataframe 타입으로 변환
# 이유: pandas는 시계열 자료에 대한 다양한 기능을 제공하고
#       LSTM에서 사용하는 window를 만들기 유용

train_sc_df = pd.DataFrame(train_sc, columns=['trade_price_idx_value'], index=train.index)
test_sc_df = pd.DataFrame(test_sc, columns=['trade_price_idx_value'], index=test.index)
train_sc_df.head()
```

In [ ]:

```
# Sliding window 구성하기
# window는 LSTM을 훈련하기 위한 단위로 고정된 사이즈를 가짐
# window가 12개라면 과거 시간 데이터 12개를 사용해서 다음 시간 단위의 값을 예측
#
# pandas의 dataframe의 shift 연산 사용

for s in range(1, 13):
    train_sc_df['shift_{}'.format(s)] = train_sc_df['trade_price_idx_value'].shift(s)
    test_sc_df['shift_{}'.format(s)] = test_sc_df['trade_price_idx_value'].shift(s)

train_sc_df.head(17)
```

In [ ]:

```
# 훈련 데이터와 시험 데이터 만들기
# dropna로 NaN 값을 포함한 데이터는 제거
# shift_1 ~ shift_12는 X로, trade_price_idx_value는 Y로 지정

X_train = train_sc_df.dropna().drop('trade_price_idx_value', axis=1)
y_train = train_sc_df.dropna()[['trade_price_idx_value']]

X_test = test_sc_df.dropna().drop('trade_price_idx_value', axis=1)
y_test = test_sc_df.dropna()[['trade_price_idx_value']]

X_train.head()
```

In [ ]:

```
y_train.head()
```

In [ ]:

```
# 다시 numpy 배열로 변환하기

print(type(X_train))
X_train = X_train.values
print(type(X_train))
X_test= X_test.values

y_train = y_train.values
y_test = y_test.values

print(X_train.shape)
print(y_train.shape)
```

In [ ]:

```
# 최종 훈련 데이터와 시험 데이터 세트의 X 만들기
# 케라스에서는 RNN 계열의 모델을 훈련할 때 요구되는 데이터 형식이 있음
# 3차원 데이터, 각각의 차원은 (size, timestep, feature)을 순서대로 나타내야 함
# 따라서 이 형태로 데이터를 reshape 해주어야 함

X_train_t = X_train.reshape(X_train.shape[0], 12, 1)
X_test_t = X_test.reshape(X_test.shape[0], 12, 1)

print("최종 DATA")
print(X_train_t.shape)
print(X_train_t)
print(y_train)
```

In [ ]:

```
# LSTM 모델 만들기

from keras.layers import LSTM
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import EarlyStopping

model = Sequential()
# 20 메모리 셀을 가진 LSTM 레이어 추가
model.add(LSTM(20, input_shape=(12, 1)))
model.add(Dense(1))

model.summary()
```

In [ ]:

```
# 학습과정 설정 및 훈련

model.compile(loss='mean_squared_error', optimizer='adam')

early_stop = EarlyStopping(monitor='loss', patience=1, verbose=1)

model.fit(X_train_t, y_train, epochs=100,
          batch_size=30, verbose=1, callbacks=[early_stop])
```

In [ ]:

```
score = model.evaluate(X_test_t, y_test, batch_size=30)
print(score)
```

In [ ]:

```
print(X_test_t)
```

In [ ]:

```
y_pred = model.predict(X_test_t)
print(y_pred)
```

In [ ]:

```
print(y_test)
```