In [1]:

```
import numpy as np
import pandas as pd
from statsmodels.tsa.api import SimpleExpSmoothing
import matplotlib.pyplot as plt
```

Time series forecasting workflow

In [2]:

```
# read dataframe saved earlier using df.to_pickle('AlgerianExport.pkl')
df=pd.read_pickle('AlgerianExport.pkl')
```

In [3]:

```
df.head(5)
```

Out[3]:

	Export
1960-12-31	39.043173
1961-12-31	46.244557
1962-12-31	19.793873
1963-12-31	24.684682
1964-12-31	25.084059

In [4]:

```
df.size
```

Out[4]:

```
58
```

Train test split

In [5]:

```
ncut=int(0.8*df.size) # 80% for training the rest is withheld for testing
ncut
```

Out[5]:

```
46
```

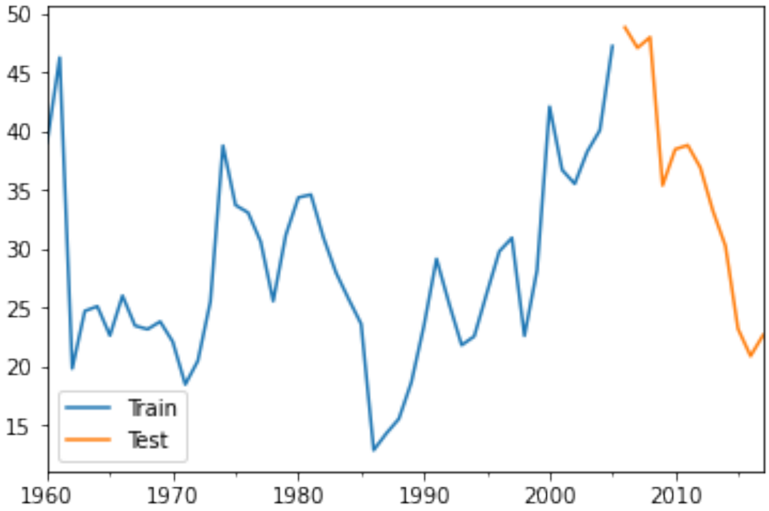
In [6]:

```
train_data=df.iloc[:ncut]
test_data=df.iloc[ncut:]
```

In [7]:

```
ax=train_data.plot()
test_data.plot(ax=ax)
ax.legend(['Train','Test'])
```

Out[7]:
```
<matplotlib.legend.Legend at 0x7f940d769130>
```



Fitting the model

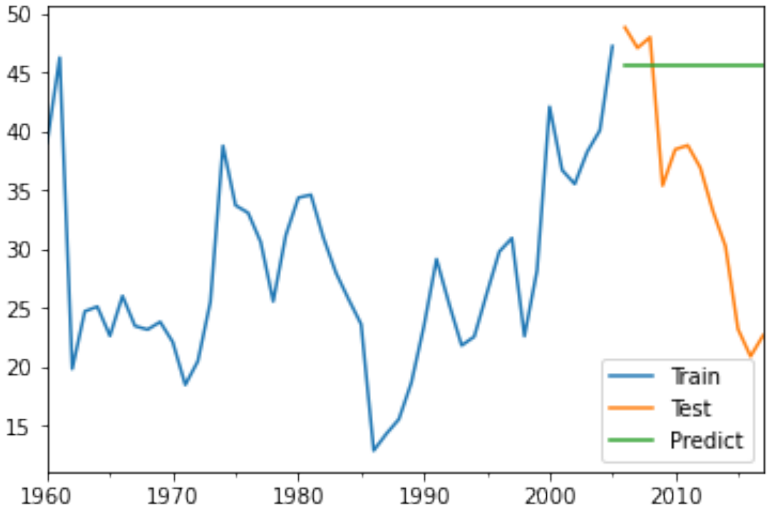
In [8]:
```
fitted_model=SimpleExpSmoothing(train_data,initialization_method='estimatec
```

In [9]:
```
test_predictions=fitted_model.forecast(test_data.size).rename('SES forecast
```

Evaluating the model against test data

In [10]:
```
ax=train_data.plot()
test_data.plot(ax=ax)
test_predictions.plot(ax=ax)
ax.legend(['Train','Test','Predict'])
```

Out[10]:
```
<matplotlib.legend.Legend at 0x7f940d8c75e0>
```



Evaluation metrics

In [11]:

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import  mean_absolute_error,mean_absolute_percentage_e
```

In [12]:

```
# MSE
mean_squared_error(test_data,test_predictions)
```

Out[12]:

```
192.04697481843695
```

In [13]:

```
# check MSE function
np.sum(np.square(np.subtract(test_data["Export"].values,test_predictions.va
```

Out[13]:

```
192.04697481843695
```

In [14]:

```
# MAE
mean_absolute_error(test_data,test_predictions)
```

Out[14]:

```
11.439488025448926
```

In [15]:

```
# MAPE # you must multiply by 100 to get percentage -check!
mean_absolute_percentage_error(test_data,test_predictions)
```

Out[15]:

```
0.42166571730854924
```

In [16]:

```
np.sum(np.divide(np.abs(np.subtract(test_data["Export"].values,test_predict
```

Out[16]:

```
0.42166571730854924
```

Forecasting into future

Let us assume that the model presented above turned out to be the best. We train the final model using all available data points.

In [17]:

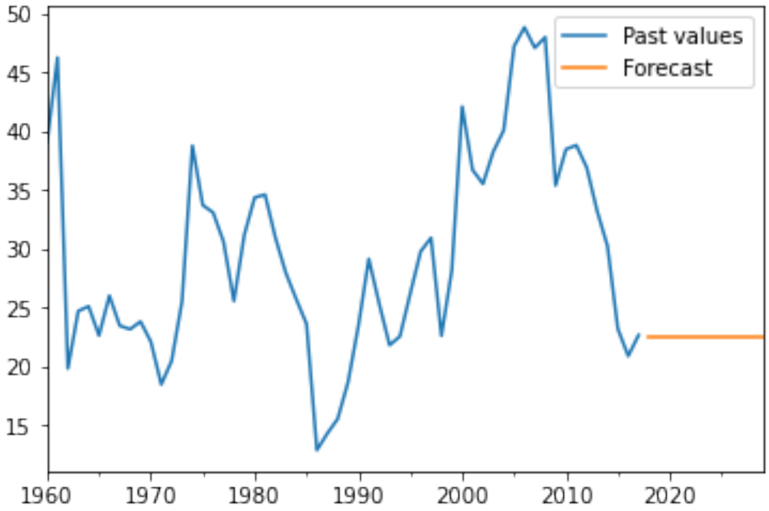
```
# we use all available data
final_model=SimpleExpSmoothing(df,initialization_method='estimated').fit()
forecast=final_model.forecast(test_data.size).rename('forecast')
```

In [18]:

```
ax=df.plot()
forecast.plot(ax=ax)
ax.legend(['Past values','Forecast'])
```

Out[18]:

```
<matplotlib.legend.Legend at 0x7f940de43f10>
```


Past values
Forecast
50
45
40
35
30
25
20
15
1960
1970
1980
1990
2000
2010
2020