In [2]:

```
import numpy as np
import pandas as pd
import scipy.stats as sp
import seaborn as sns
import matplotlib.pyplot as plt
from pandas.plotting import register_matplotlib_converters
from sklearn.linear_model import LinearRegression as lm
import statsmodels.api as sm
```

In [3]:

```
def difference(dataset,interval=1):
    diff=[]
    for i in range(interval,len(dataset)):
        value=dataset[i]-dataset[i-interval]
        diff.append(value)
    return diff
```

In [4]:

```
def runMyAR1(yin):
    tlen = len(yin)
    y = np.array(yin[2:tlen])
    x = np.array(yin[1:(tlen-1)])
    X = x
    X = sm.add_constant(X)
    regr2 = sm.OLS(y,X)
    model = regr2.fit()
    print(model.summary())
    ypred = model.predict()
    plt.plot((y-ypred))
```

In [5]:

```
# Create large images!
register_matplotlib_converters()
sns.set_style("darkgrid")
plt.rc("figure", figsize=(14, 8)) # was 16,12
plt.rc("font", size=13)
```

In [6]:

```
dfo = pd.read_csv("Coal Power.csv")
#dfo.columns = ['Month','Price']
dfo.head()
```

Out[6]:

	Unnamed: 0	Total consumption : Texas : electric power (total) : quarterly (short tons)
0	2001 Q1	22164839
1	2001 Q2	22952510
2	2001 Q3	25962808
3	2001 Q4	21357650
4	2002 Q1	21917084

In [7]:

```
dfo.rename(columns={"Unnamed: 0":"Period","Total consumption : Texas : electric power
```

In [8]:

```
dfo.head()
```

Out[8]:

	Period	E-Power
0	2001 Q1	22164839
1	2001 Q2	22952510
2	2001 Q3	25962808
3	2001 Q4	21357650
4	2002 Q1	21917084

In [9]:

```
df = dfo['E-Power']
print(len(df))
```

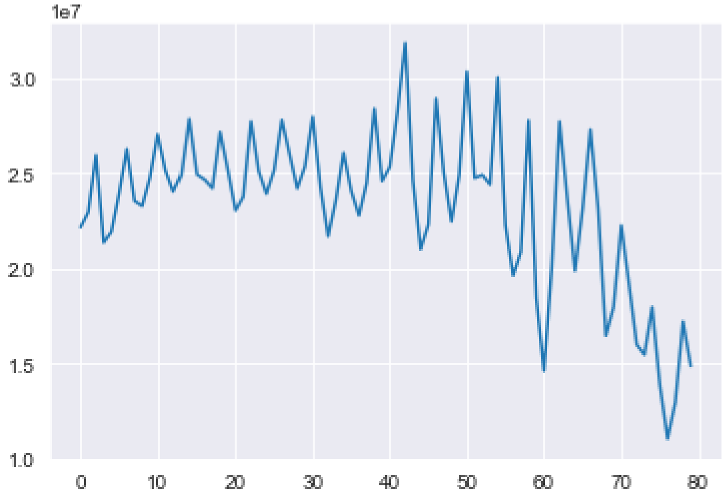
```
80
```

In [10]:

```
df.plot()
```

Out[10]:

```
<AxesSubplot:>
```



In [11]:

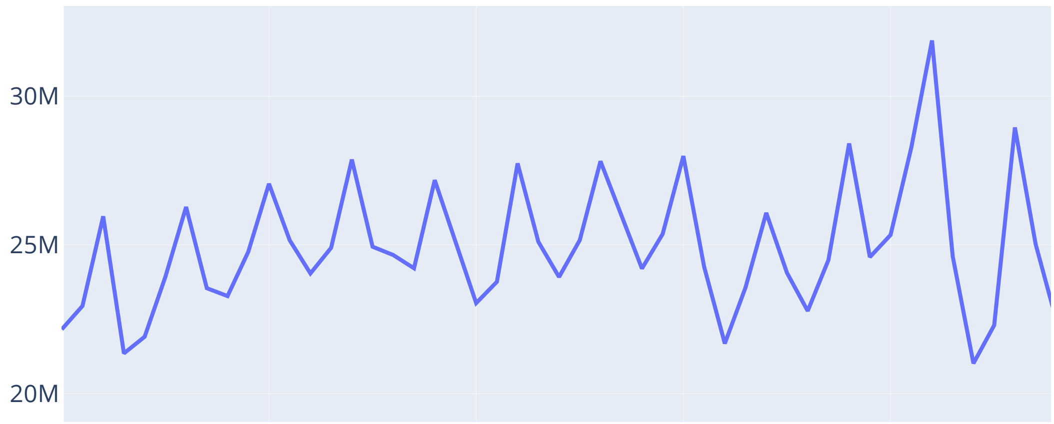
```
#import chart_studio.plotly as py
import plotly.graph_objs as go
# Offline mode
import plotly.io as pio
#from plotly.offline import init_notebook_mode, iplot
#init_notebook_mode(connected=True)
```

In [12]:

```
#pltobj = go.Scatter(y=dfo['Price'], x=dfo['Month'])
pltobj = go.Scatter(y=dfo['E-Power'])
```

In [13]:

```
fig = go.Figure(data=pltobj)
pio.show(fig)
```



In [14]:

```
sm.tsa.stattools.adfuller(df)
```

Out[14]:

```
(1.9763029096941296,
 0.9986413567364828,
 10,
 69,
 {'1%': -3.528889992207215,
  '5%': -2.9044395987933362,
  '10%': -2.589655654274312},
 2117.1250184879855)
```

In [15]:

```
sm.tsa.stattools.kpss(df)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\tsa\stattools.py:2018: InterpolationWarning:

The test statistic is outside of the range of p-values available in the
look-up table. The actual p-value is smaller than the p-value returned.
```

Out[15]:

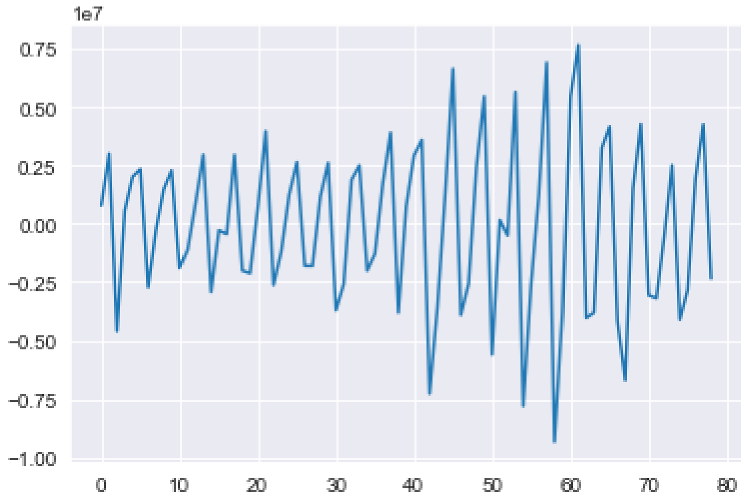
```
(0.8419408149624352,
 0.01,
 4,
 {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739})
```

In [16]:

```
# detrend, if required, and plot
dtrend = difference(np.array(df),1)
#dtrend = df
plt.plot(dtrend)
```

Out[16]:

```
[<matplotlib.lines.Line2D at 0x26d80060eb0>]
```



In [17]:

```
sm.tsa.stattools.adfuller(dtrend)
```

Out[17]:

```
(-4.510983475470285,
 0.0001878466316890073,
 5,
 73,
 {'1%': -3.5232835753964475,
  '5%': -2.902030597326081,
  '10%': -2.5883710883843123},
 2087.3684260623113)
```

In [18]:

```
sm.tsa.stattools.kpss(dtrend)
```

Out[18]:

```
(0.4356019919994831,
 0.061809486207119374,
 15,
 {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739})
```

In [19]:

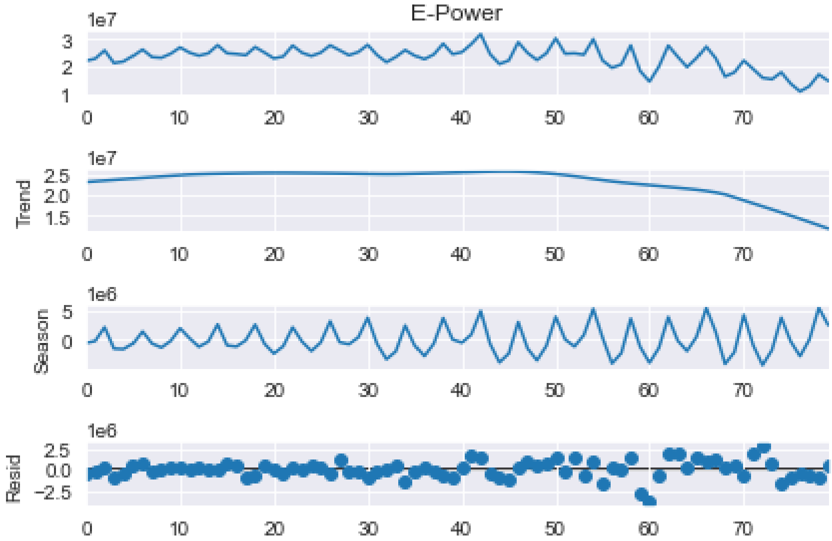
```
from statsmodels.tsa.seasonal import STL
```

In [20]:

```
stl = STL(df, period=12)
#stl = STL(df)
```

In [21]:

```
res = stl.fit()
fig = res.plot()
```



ARIMA MODEL

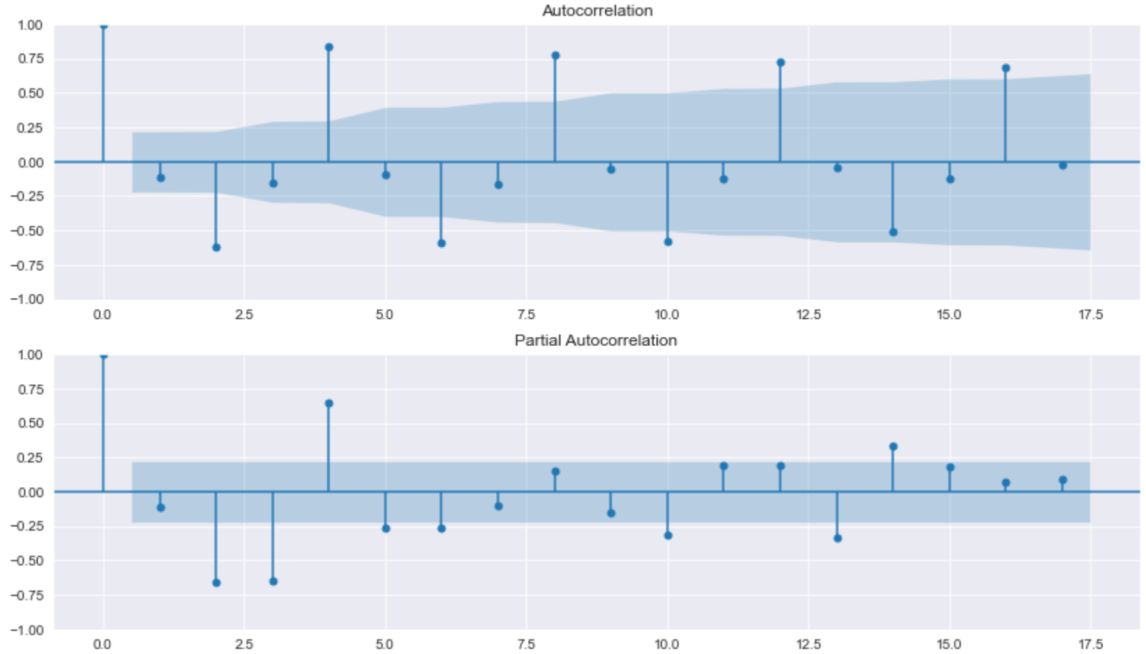
In [22]:

```
import statsmodels.api as sm
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf
```

In [23]:

```
fig = plt.figure(figsize=(14, 8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(dtrend, lags=17, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(dtrend, lags=17, ax=ax2)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\graphics\tsaplots.py:348: FutureWarning:

The default method 'yw' can produce PACF values outside of the [-1,1] i
nterval. After 0.13, the default will change tounadjusted Yule-Walker
('ywm'). You can use this method now by setting method='ywm'.
```



In [24]:

```
sm.tsa.stattools.arma_order_select_ic(dtrend)
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\tsa\statespace\sarimax.py:978: UserWarning:

Non-invertible starting MA parameters found. Using zeros as starting pa
rameters.
```

Out[24]:

```
{'bic':               0            1            2
 0  2883.069229  2599.693050  2600.513358
 1  2619.240609  2600.665017  2604.347110
 2  2580.555328  2575.732558  2555.417793
 3  2548.075730  2529.529455  2516.342855
 4  2513.901381  2515.133141  2515.470113,
 'bic_min_order': (4, 0)}
```

In [25]:

```
from statsmodels.tsa.arima.model import ARIMA
```

In [26]:

```
my_model = sm.tsa.arima.ARIMA(df,order=(4,1,0),seasonal_order=(4,1,0,12))
my_model_res = my_model.fit()
print(my_model_res.summary())
```

```
                                     SARIMAX Results                                      
=========================================================================
=================
Dep. Variable:                            E-Power   No. Observations:                   
80
Model:             ARIMA(4, 1, 0)x(4, 1, 0, 12)   Log Likelihood                
-1065.876
Date:                          Mon, 20 Feb 2023   AIC                           
2149.752
Time:                                  19:19:09   BIC                           
2169.594
Sample:                                       0   HQIC                          
2157.603
                                           - 80                                         
Covariance Type:                            opg                                         
=========================================================================
=======
                 coef    std err          z      P>|z|      [0.025      
0.975]
-------------------------------------------------------------------------
-------
ar.L1         -0.0510      0.118     -0.433      0.665      -0.282      
0.180
ar.L2         -0.0914      0.078     -1.175      0.240      -0.244      
0.061
ar.L3         -0.1802      0.072     -2.507      0.012      -0.321      
-0.039
ar.L4          0.1620      0.071      2.268      0.023       0.022      
0.302
ar.S.L12      -0.2379      0.085     -2.800      0.005      -0.404      
-0.071
ar.S.L24      -0.1300      0.094     -1.386      0.166      -0.314      
0.054
ar.S.L36      -0.0123      0.104     -0.118      0.906      -0.217      
0.192
ar.S.L48      -0.0026      0.077     -0.034      0.973      -0.153      
0.148
sigma2      3.843e+12   8.44e-15   4.55e+26      0.000    3.84e+12    
3.84e+12
=========================================================================
============
Ljung-Box (L1) (Q):                   0.95   Jarque-Bera (JB):                 
0.54
Prob(Q):                              0.33   Prob(JB):                         
0.76
Heteroskedasticity (H):               3.12   Skew:                             
-0.16
Prob(H) (two-sided):                  0.01   Kurtosis:                         
2.70
=========================================================================
============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients 
(complex-step).
[2] Covariance matrix is singular or near-singular, with condition numb
er 9.69e+41. Standard errors may be unstable.
```

In [27]:

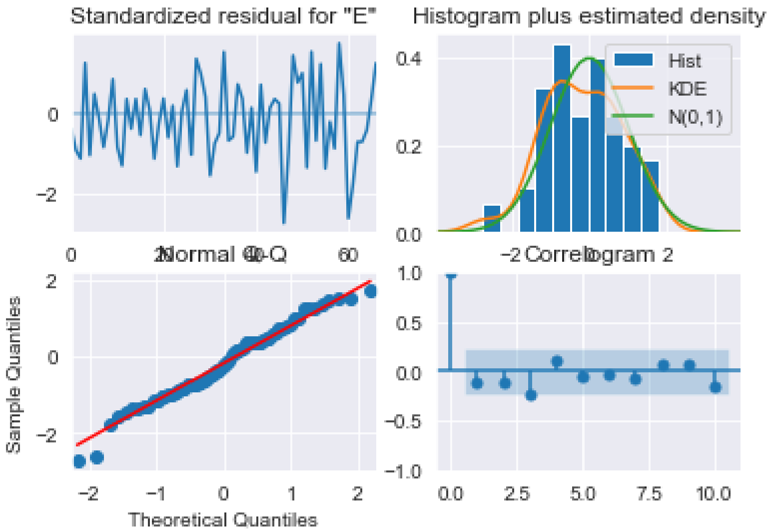
```
type(my_model_res)
```

Out[27]:

```
statsmodels.tsa.arima.model.ARIMAResultsWrapper
```

In [28]:

```
pred = my_model_res.plot_diagnostics()
```



In [29]:

```
tforecast =  my_model_res.forecast(24)
tforecast2 = my_model_res.get_forecast(24)
confint = np.array(tforecast2.conf_int())
```

In [30]:

```
type(confint)
```

Out[30]:

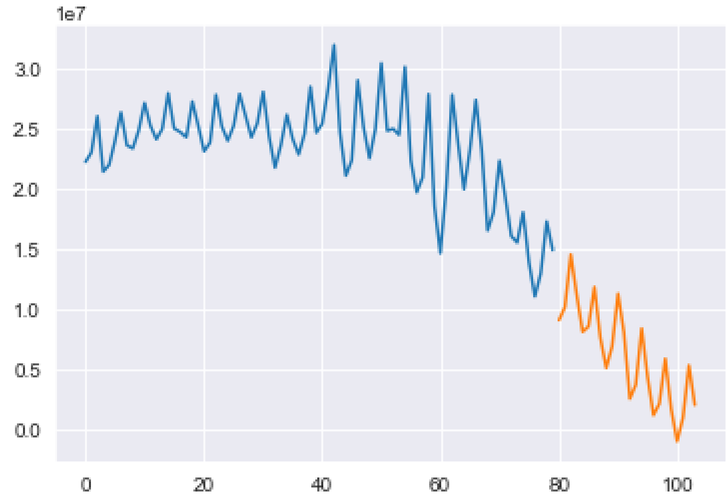
```
numpy.ndarray
```

In [31]:

```
plt.plot(df)
plt.plot(tforecast)
```

Out[31]:

```
[<matplotlib.lines.Line2D at 0x26d80542da0>]
```

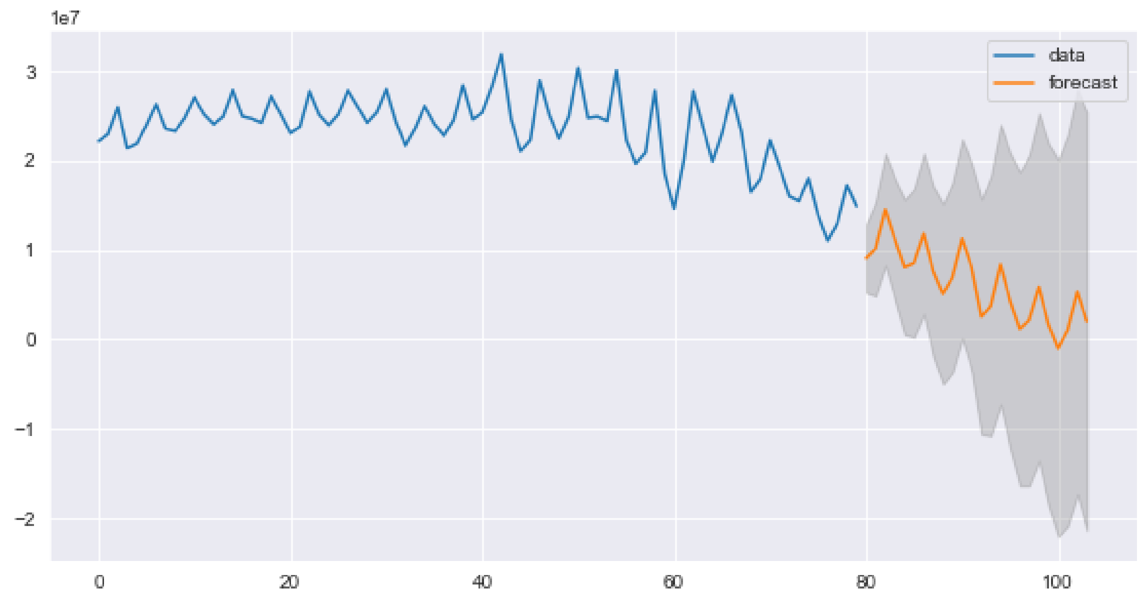


In [32]:

```
fig,ax = plt.subplots(figsize=(10,5))
ax.plot(df.index, df, label='data')
ax.plot(tforecast2.predicted_mean.index, tforecast2.predicted_mean, label='forecast')
ax.fill_between(tforecast2.predicted_mean.index, confint[:,0], confint[:,1],color='gr
ax.legend()
```

Out[32]:

```
<matplotlib.legend.Legend at 0x26d80541a50>
```



AUTO ARIMA MODEL

In [33]:

```
import pmdarima as pm
```

In [34]:

```
myfit = pm.auto_arima(df,start_p=0, start_q=0,
                             max_p=4, max_q=3, m=12,
                             start_P=0, seasonal=True,
                             d=1, D=1, trace=True,
                             error_action='ignore',  # don't want to know if an order
                             suppress_warnings=True,  # don't want convergence warnin
                             stepwise=True)  # set to stepwise
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,1,1)[12]             : AIC=2156.159, Time=0.24 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=2151.809, Time=0.04 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=2152.071, Time=0.19 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=2151.137, Time=0.23 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=2151.990, Time=0.07 sec
 ARIMA(0,1,1)(1,1,1)[12]             : AIC=2152.210, Time=0.57 sec
 ARIMA(0,1,1)(0,1,2)[12]             : AIC=2151.548, Time=0.69 sec
 ARIMA(0,1,1)(1,1,0)[12]             : AIC=2151.921, Time=0.17 sec
 ARIMA(0,1,1)(1,1,2)[12]             : AIC=2153.312, Time=0.99 sec
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=2152.028, Time=0.62 sec
 ARIMA(0,1,2)(0,1,1)[12]             : AIC=2151.969, Time=0.30 sec
 ARIMA(1,1,0)(0,1,1)[12]             : AIC=2151.313, Time=0.17 sec
 ARIMA(1,1,2)(0,1,1)[12]             : AIC=2153.393, Time=0.56 sec
 ARIMA(0,1,1)(0,1,1)[12] intercept   : AIC=2152.160, Time=0.26 sec

Best model:  ARIMA(0,1,1)(0,1,1)[12]
Total fit time: 5.118 seconds
```

WINTER-HOLTS MODEL

In [35]:

```
rolling = dfo['E-Power'].rolling(12)
type(rolling)
```

Out[35]:

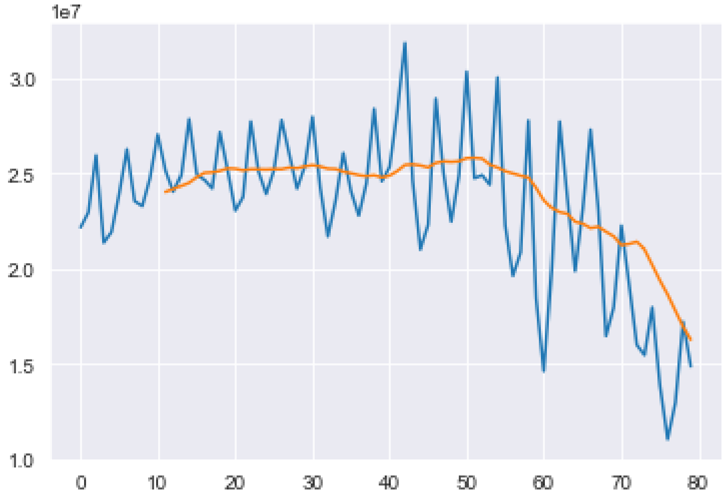
```
pandas.core.window.rolling.Rolling
```

In [36]:

```
mav = rolling.mean()
plt.plot(dfo['E-Power'])
plt.plot(mav)
```

Out[36]:

```
[<matplotlib.lines.Line2D at 0x26d869f5600>]
```



In [37]:

```
type(mav)
```

Out[37]:

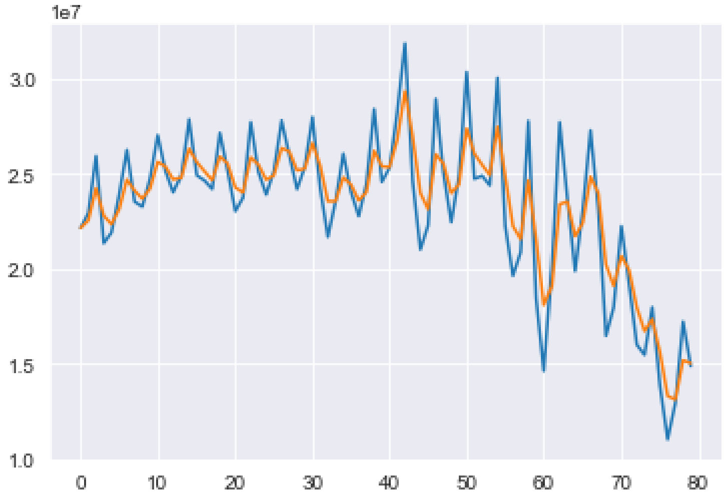
```
pandas.core.series.Series
```

In [38]:

```
# Try out the following with various values of 'alpha' and evaluate the results
ewma = dfo['E-Power'].ewm(alpha=0.5, adjust=False).mean()
plt.plot(dfo['E-Power'])
plt.plot(ewma)
```

Out[38]:

```
[<matplotlib.lines.Line2D at 0x26d86a5aef0>]
```

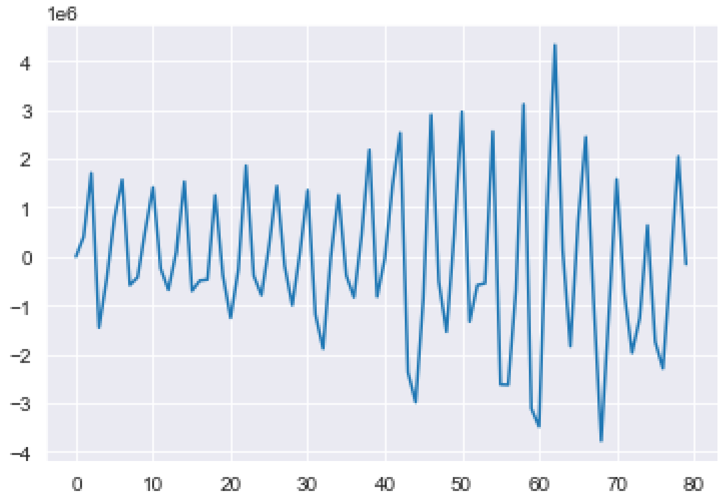


In [39]:

```
plt.plot(dfo['E-Power']-ewma)
```

Out[39]:

```
[<matplotlib.lines.Line2D at 0x26d88abdff0>]
```



In [40]:

```
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
```

In [41]:

```
ses = SimpleExpSmoothing(df)
```

In [42]:

```
type(ses)
```

Out[42]:

```
statsmodels.tsa.holtwinters.model.SimpleExpSmoothing
```

In [43]:

```
result = ses.fit(smoothing_level=0.1, optimized=False)
```

In [44]:

```
result.summary()
```

Out[44]:

SimpleExpSmoothing Model Results

Dep. Variable:	E-Power	**No. Observations:**	80
Model:	SimpleExpSmoothing	**SSE**	915218536886908.250
Optimized:	False	**AIC**	2409.453
Trend:	None	**BIC**	2414.217
Seasonal:	None	**AICC**	2409.986
Seasonal Periods:	None	**Date:**	Mon, 20 Feb 2023
Box-Cox:	False	**Time:**	19:19:26
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.1000000	alpha	False
initial_level	2.2165e+07	l.0	False

In [45]:

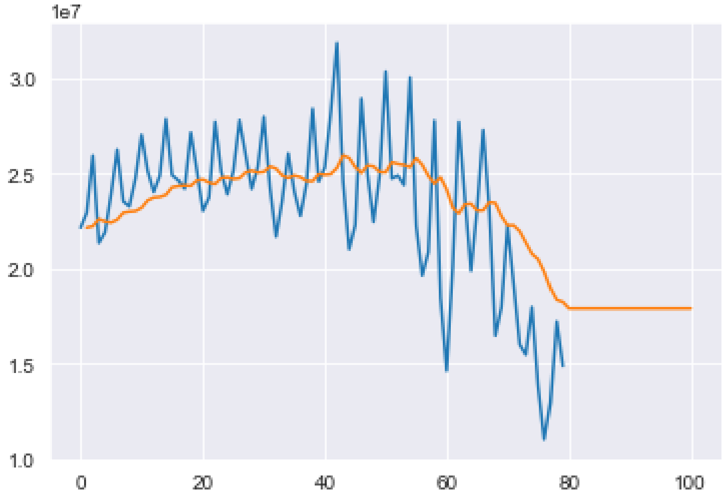
```
mypred = result.predict(start=1, end=100)
```

In [46]:

```
plt.plot(df)
plt.plot(mypred)
```

Out[46]:

```
[<matplotlib.lines.Line2D at 0x26d8da61030>]
```



In [47]:

```
result.params
```

Out[47]:

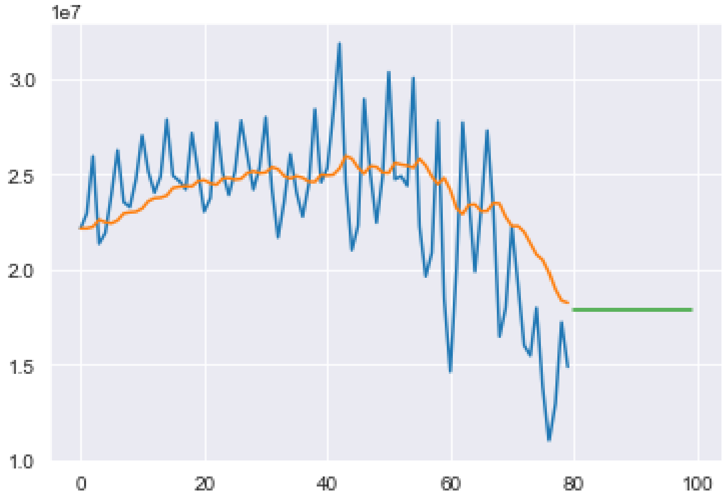
```
{'smoothing_level': 0.1,
 'smoothing_trend': None,
 'smoothing_seasonal': None,
 'damping_trend': nan,
 'initial_level': 22164839.0,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [48]:

```
plt.plot(df)
plt.plot(result.fittedvalues)
plt.plot(result.forecast(20))
```

Out[48]:

```
[<matplotlib.lines.Line2D at 0x26d8dacc2e0>]
```



In [49]:

```
result2 = ses.fit() # optimize the values of alpha
```

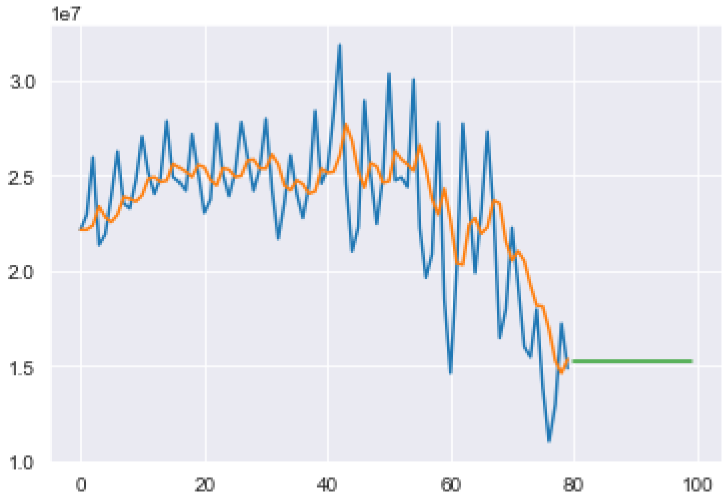
```
C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\tsa\holtwinters\model.py:915: ConvergenceWarning:

Optimization failed to converge. Check mle_retvals.
```

In [50]:

```
plt.plot(df)
plt.plot(result2.fittedvalues)
plt.plot(result2.forecast(20))
```

Out[50]:

```
[<matplotlib.lines.Line2D at 0x26d8db17370>]
```



In [51]:

```
result2.params
```

Out[51]:

```
{'smoothing_level': 0.2812790697674419,
 'smoothing_trend': nan,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 22164839.0,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [52]:

```
from statsmodels.tsa.holtwinters import Holt
```

In [53]:

```
model = Holt(df, exponential=True)
result = model.fit()
result.params
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\tsa\holtwinters\model.py:915: ConvergenceWarning:

Optimization failed to converge. Check mle_retvals.
```

Out[53]:

```
{'smoothing_level': 0.18837330823548065,
 'smoothing_trend': 0.18331477657668624,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 21714307.6637008,
 'initial_trend': 1.007728898869419,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [54]:

```
result.summary()
```

Out[54]:

Holt Model Results

Dep. Variable:	E-Power	**No. Observations:**	80
Model:	Holt	**SSE**	766833837996925.625
Optimized:	True	**AIC**	2399.301
Trend:	Multiplicative	**BIC**	2408.829
Seasonal:	None	**AICC**	2400.452
Seasonal Periods:	None	**Date:**	Mon, 20 Feb 2023
Box-Cox:	False	**Time:**	19:19:29
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.1883733	alpha	True
smoothing_trend	0.1833148	beta	True
initial_level	2.1714e+07	l.0	True
initial_trend	1.0077289	b.0	True

In [55]:

```
plt.plot(df)
plt.plot(result.fittedvalues)
plt.plot(result.forecast(20))
```

Out[55]:

```
[<matplotlib.lines.Line2D at 0x26d8021ff70>]
```

In [56]:

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

In [57]:

```
model = ExponentialSmoothing(df, trend='mul', seasonal='mul', seasonal_periods=12)
```

In [58]:

```
result3 = model.fit()
result3.params
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\tsa\holtwinters\model.py:915: ConvergenceWarning:

Optimization failed to converge. Check mle_retvals.
```

Out[58]:

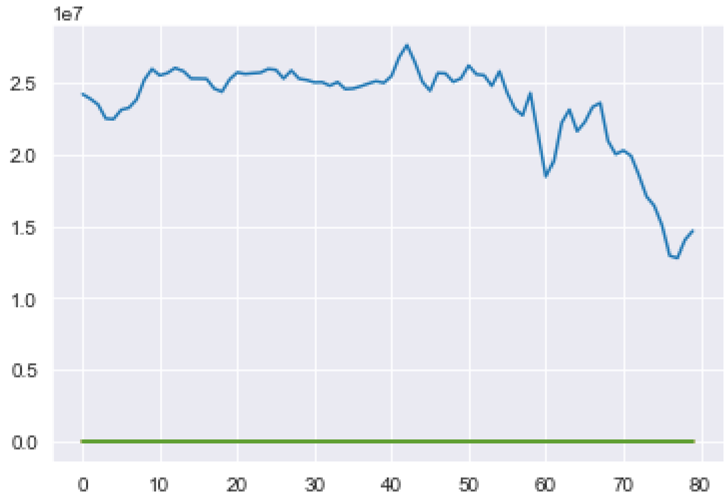
```
{'smoothing_level': 0.5353571428571429,
 'smoothing_trend': 0.025493197278911566,
 'smoothing_seasonal': 0.0001,
 'damping_trend': nan,
 'initial_level': 24082946.07222221,
 'initial_trend': 1.0055785269054447,
 'initial_seasons': array([0.91797085, 0.97842296, 1.12838805, 0.991717
93, 0.98135232,
        1.01447578, 1.12906098, 0.97248173, 0.88556166, 0.93586909,
        1.08454598, 0.98015267]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [59]:

```
plt.plot(result3.level)
plt.plot(result3.trend)
plt.plot(result3.season)
```

Out[59]:

```
[<matplotlib.lines.Line2D at 0x26d802a5d50>]
```



In [60]:

```
result3.summary()
```

Out[60]:

ExponentialSmoothing Model Results

Dep. Variable:	E-Power	**No. Observations:**	80
Model:	ExponentialSmoothing	**SSE**	266642232073999.000
Optimized:	True	**AIC**	2338.792
Trend:	Multiplicative	**BIC**	2376.905
Seasonal:	Multiplicative	**AICC**	2350.005
Seasonal Periods:	12	**Date:**	Mon, 20 Feb 2023
Box-Cox:	False	**Time:**	19:19:31
Box-Cox Coeff.:	None		

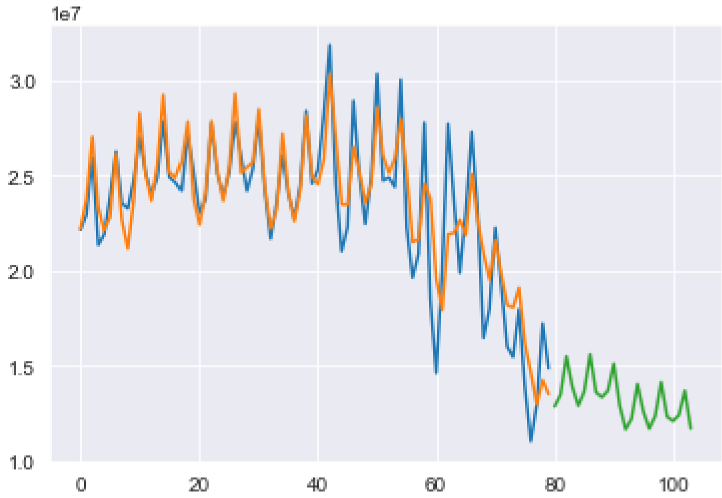
	coeff	code	optimized
smoothing_level	0.5353571	alpha	True
smoothing_trend	0.0254932	beta	True
smoothing_seasonal	0.0001	gamma	True
initial_level	2.4083e+07	l.0	True
initial_trend	1.0055785	b.0	True
initial_seasons.0	0.9179709	s.0	True
initial_seasons.1	0.9784230	s.1	True
initial_seasons.2	1.1283880	s.2	True
initial_seasons.3	0.9917179	s.3	True
initial_seasons.4	0.9813523	s.4	True
initial_seasons.5	1.0144758	s.5	True
initial_seasons.6	1.1290610	s.6	True
initial_seasons.7	0.9724817	s.7	True
initial_seasons.8	0.8855617	s.8	True
initial_seasons.9	0.9358691	s.9	True
initial_seasons.10	1.0845460	s.10	True
initial_seasons.11	0.9801527	s.11	True

In [61]:

```
plt.plot(df)
plt.plot(result3.fittedvalues)
plt.plot(result3.forecast(24))
```

Out[61]:

```
[<matplotlib.lines.Line2D at 0x26d802782e0>]
```



In [62]:

```
model2 = ExponentialSmoothing(df, trend='add', seasonal='mul', seasonal_periods=12)
```

In [63]:

```
result4 = model2.fit()
result4.params
```

```
C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\tsa\holtwinters\model.py:915: ConvergenceWarning:

Optimization failed to converge. Check mle_retvals.
```

Out[63]:

```
{'smoothing_level': 0.5353571428571429,
 'smoothing_trend': 0.025493197278911566,
 'smoothing_seasonal': 0.0001,
 'damping_trend': nan,
 'initial_level': 24082946.07222221,
 'initial_trend': 134347.36262626387,
 'initial_seasons': array([0.91797085, 0.97842296, 1.12838805, 0.991717
93, 0.98135232,
        1.01447578, 1.12906098, 0.97248173, 0.88556166, 0.93586909,
        1.08454598, 0.98015267]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [64]:

```
result4.summary()
```

Out[64]:

ExponentialSmoothing Model Results

Dep. Variable:	E-Power	**No. Observations:**	80
Model:	ExponentialSmoothing	**SSE**	266092287431320.188
Optimized:	True	**AIC**	2338.627
Trend:	Additive	**BIC**	2376.739
Seasonal:	Multiplicative	**AICC**	2349.840
Seasonal Periods:	12	**Date:**	Mon, 20 Feb 2023
Box-Cox:	False	**Time:**	19:19:34
Box-Cox Coeff.:	None		

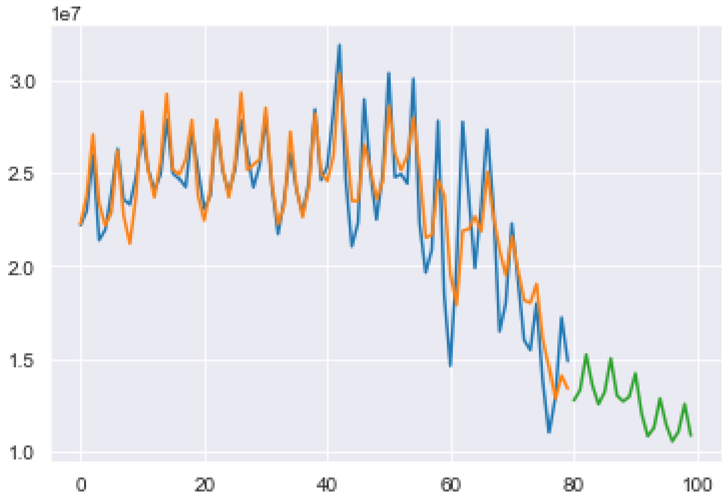
	coeff	code	optimized
smoothing_level	0.5353571	alpha	True
smoothing_trend	0.0254932	beta	True
smoothing_seasonal	0.0001	gamma	True
initial_level	2.4083e+07	l.0	True
initial_trend	1.3435e+05	b.0	True
initial_seasons.0	0.9179709	s.0	True
initial_seasons.1	0.9784230	s.1	True
initial_seasons.2	1.1283880	s.2	True
initial_seasons.3	0.9917179	s.3	True
initial_seasons.4	0.9813523	s.4	True
initial_seasons.5	1.0144758	s.5	True
initial_seasons.6	1.1290610	s.6	True
initial_seasons.7	0.9724817	s.7	True
initial_seasons.8	0.8855617	s.8	True
initial_seasons.9	0.9358691	s.9	True
initial_seasons.10	1.0845460	s.10	True
initial_seasons.11	0.9801527	s.11	True

In [65]:

```
plt.plot(df)
plt.plot(result4.fittedvalues)
plt.plot(result4.forecast(20))
```

Out[65]:

```
[<matplotlib.lines.Line2D at 0x26d8eb41e40>]
```



In [66]:

```
plt.plot(result4.fittedvalues)
```

Out[66]:

```
[<matplotlib.lines.Line2D at 0x26d8eb985e0>]
```