In [8]:

```
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 [9]:

```
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 [10]:

```
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 [11]:

```
# 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 [12]:

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

Out[12]:

	Unnamed: 0	Total carbon dioxide emissions from all sectors, all fuels, California (million metric tons CO2)
0	1980	346.183721
1	1981	334.381538
2	1982	298.004398
3	1983	293.436371
4	1984	315.858105

In [13]:

```
dfo.rename(columns={"Unnamed: 0":"Year","Total carbon dioxide emissions from all sect
```

In [14]:

```
dfo.head()
```

Out[14]:

	Year	Co2 Emission
0	1980	346.183721
1	1981	334.381538
2	1982	298.004398
3	1983	293.436371
4	1984	315.858105

In [16]:

```
df = dfo['Co2 Emission']
print(len(df))
```

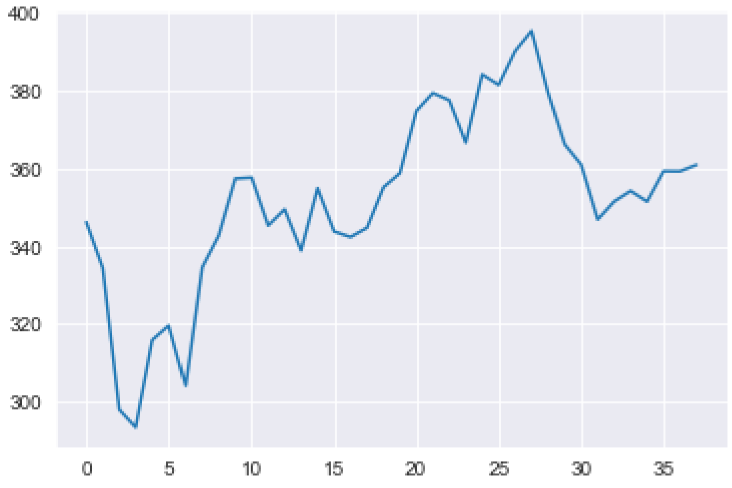
```
38
```

In [17]:

```
df.plot()
```

Out[17]:

```
<AxesSubplot:>
```



In [18]:

```
#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 [20]:

```
#pltobj = go.Scatter(y=dfo['Price'], x=dfo['Month'])
pltobj = go.Scatter(y=dfo['Co2 Emission'])
```

In [21]:

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

In [22]:

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

Out[22]:

```
(-1.6043819955568746,
 0.4813919056220212,
 0,
 37,
 {'1%': -3.6209175221605827,
  '5%': -2.9435394610388332,
  '10%': -2.6104002410518627},
 198.68496233324055)
```

In [23]:

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

Out[23]:

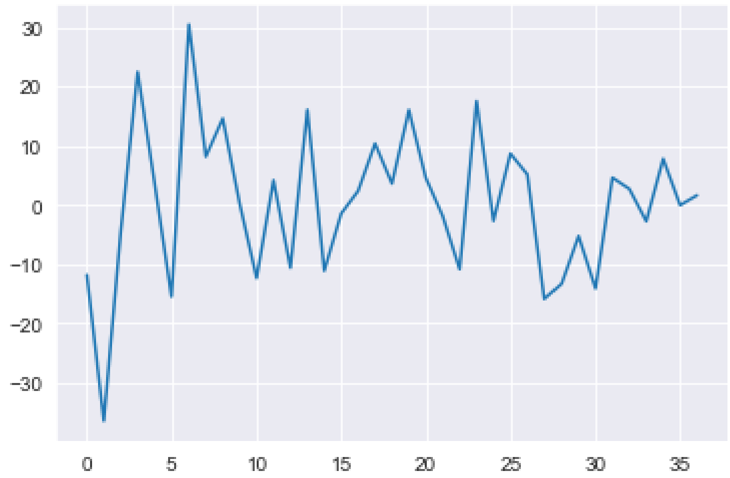
```
(0.5128755053070844,
 0.03876677808399001,
 4,
 {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739})
```

In [24]:

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

Out[24]:

```
[<matplotlib.lines.Line2D at 0x205db91d000>]
```



In [25]:

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

Out[25]:

```
(-5.7487623008790685,
 6.034785265900349e-07,
 0,
 36,
 {'1%': -3.626651907578875,
  '5%': -2.9459512825788754,
  '10%': -2.6116707716049383},
 192.82497907755067)
```

In [26]:

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

```
C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\tsa\stattools.py:2022: InterpolationWarning:

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

Out[26]:

```
(0.07569986853366702,
 0.1,
 1,
 {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739})
```

In [27]:

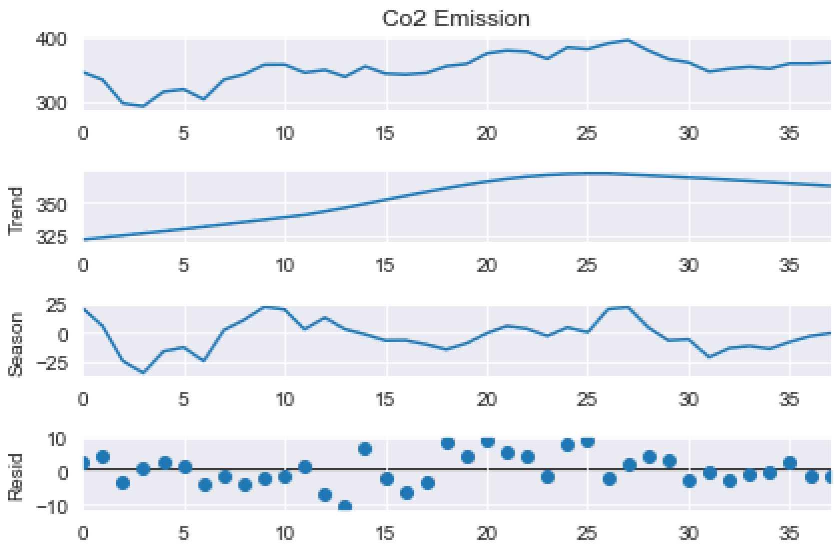
```
from statsmodels.tsa.seasonal import STL
```

In [32]:

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

In [33]:

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



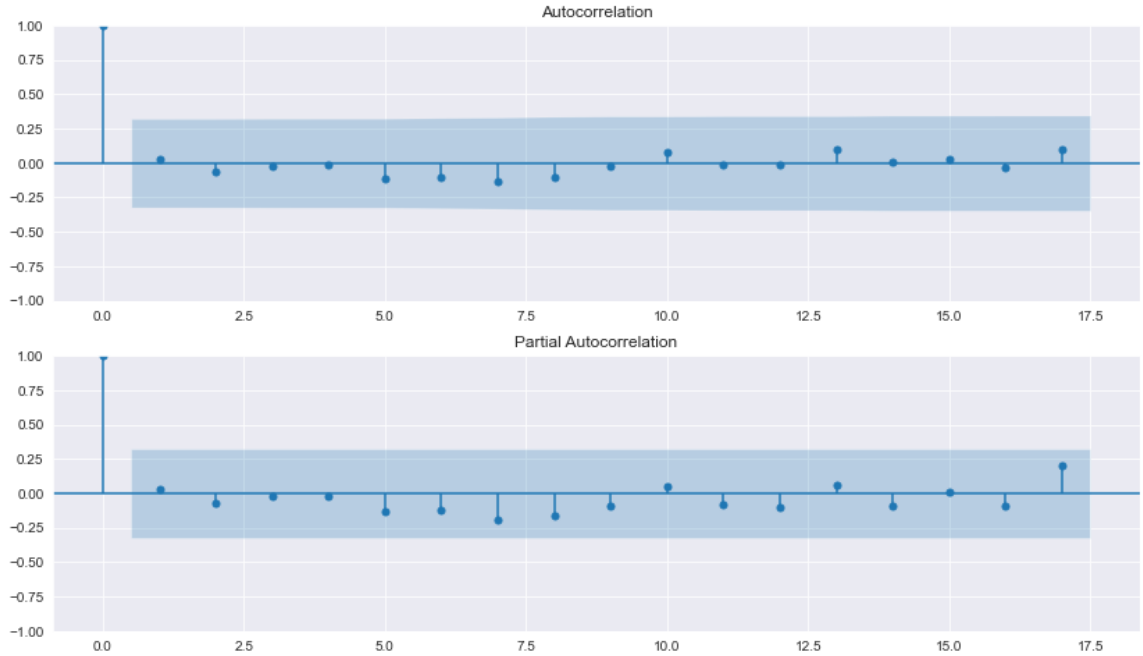
MANNUAL ARIMA

In [34]:

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

In [37]:

```
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)
```



In [38]:

```
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:966: UserWarning:

Non-stationary starting autoregressive parameters found. Using zeros as
starting parameters.

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.

C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\base\model.py:604: ConvergenceWarning:

Maximum Likelihood optimization failed to converge. Check mle_retvals

C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\base\model.py:604: ConvergenceWarning:

Maximum Likelihood optimization failed to converge. Check mle_retvals

C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\base\model.py:604: ConvergenceWarning:

Maximum Likelihood optimization failed to converge. Check mle_retvals

C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\base\model.py:604: ConvergenceWarning:

Maximum Likelihood optimization failed to converge. Check mle_retvals

C:\Users\HP\AppData\Local\Programs\Python\Python310\lib\site-packages\s
tatsmodels\base\model.py:604: ConvergenceWarning:

Maximum Likelihood optimization failed to converge. Check mle_retvals
```

Out[38]:

```
{'bic':             0           1           2
 0  299.240133  302.819466  306.243512
 1  302.823386  304.076837  307.096634
 2  306.244060  307.041606  309.283533
 3  309.827442  310.573480  314.136055
 4  313.426623  313.697105  317.605123,
 'bic_min_order': (0, 0)}
```

In [39]:

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

In [41]:

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

```
                                     SARIMAX Results
==========================================================================
=================
Dep. Variable:                    Co2 Emission   No. Observations:
38
Model:             ARIMA(0, 1, 0)x(0, 1, 0, 12)   Log Likelihood
-106.421
Date:                         Mon, 20 Feb 2023   AIC
214.842
Time:                                 11:53:26   BIC
216.061
Sample:                                      0   HQIC
215.180
                                          - 38
Covariance Type:                           opg
==========================================================================
=======
                 coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------
-------
sigma2       291.6823     62.928      4.635      0.000     168.346
415.018
==========================================================================
============
Ljung-Box (L1) (Q):                   0.38   Jarque-Bera (JB):
8.20
Prob(Q):                              0.53   Prob(JB):
0.02
Heteroskedasticity (H):               0.28   Skew:
1.12
Prob(H) (two-sided):                  0.09   Kurtosis:
4.70
==========================================================================
============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
```

In [42]:

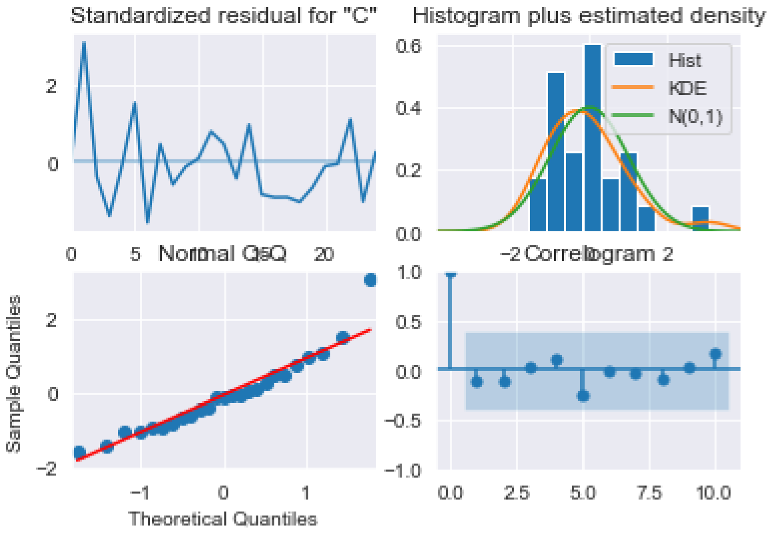
```
type(my_model_res)
```

Out[42]:

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

In [43]:

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



In [44]:

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

In [45]:

```
type(confint)
```

Out[45]:

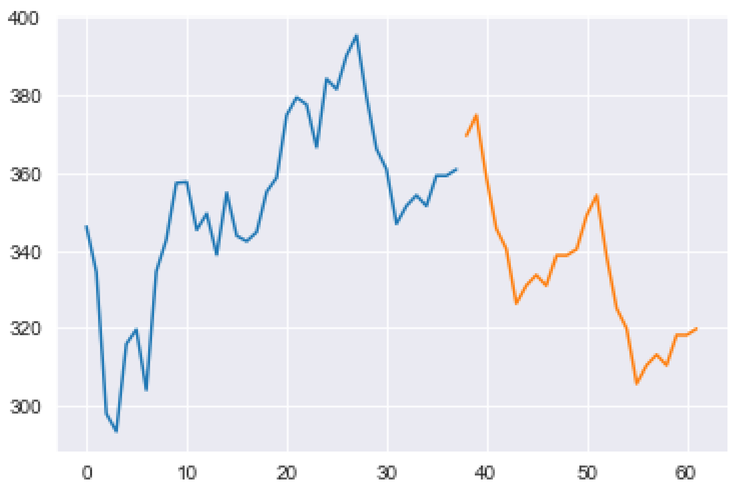
```
numpy.ndarray
```

In [47]:

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

Out[47]:

```
[<matplotlib.lines.Line2D at 0x205dbd749d0>]
```

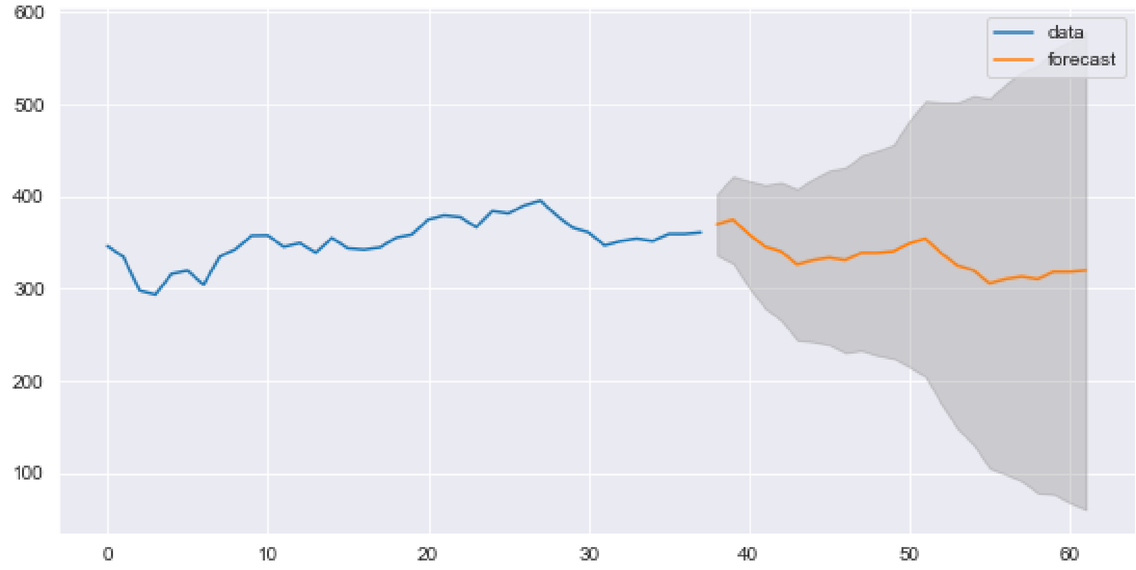


In [56]:

```
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[56]:

```
<matplotlib.legend.Legend at 0x205df8016f0>
```



AUTO ARIMA

In [57]:

```
import pmdarima as pm
```

In [58]:

```
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=216.763, Time=0.12 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=214.842, Time=0.03 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=218.512, Time=0.17 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=218.441, Time=0.22 sec
 ARIMA(0,1,0)(1,1,0)[12]             : AIC=216.763, Time=0.18 sec
 ARIMA(0,1,0)(1,1,1)[12]             : AIC=inf, Time=0.68 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=216.534, Time=0.05 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=216.453, Time=0.05 sec
 ARIMA(1,1,1)(0,1,0)[12]             : AIC=inf, Time=0.30 sec
 ARIMA(0,1,0)(0,1,0)[12] intercept   : AIC=216.763, Time=0.02 sec

Best model:  ARIMA(0,1,0)(0,1,0)[12]
Total fit time: 1.832 seconds
```

WINTER-HOLTS MODEL

In [67]:

```
rolling = dfo['Co2 Emission'].rolling(3)
type(rolling)
```

Out[67]:

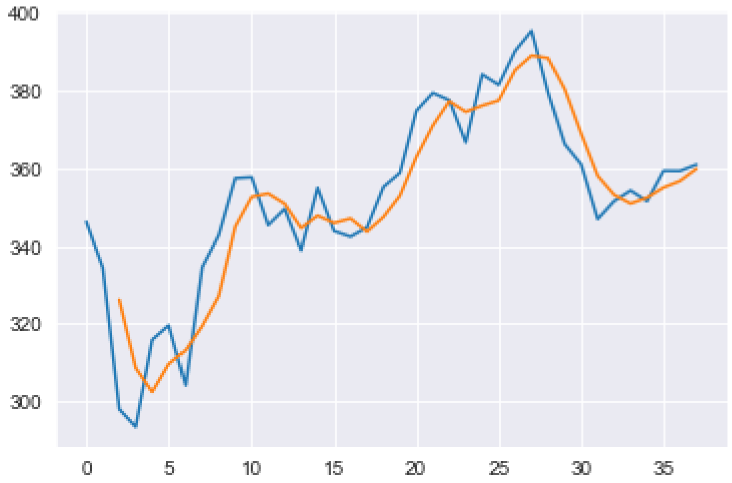
```
pandas.core.window.rolling.Rolling
```

In [68]:

```
mav = rolling.mean()
plt.plot(dfo['Co2 Emission'])
plt.plot(mav)
```

Out[68]:

```
[<matplotlib.lines.Line2D at 0x205df867910>]
```



In [69]:

```
type(mav)
```

Out[69]:

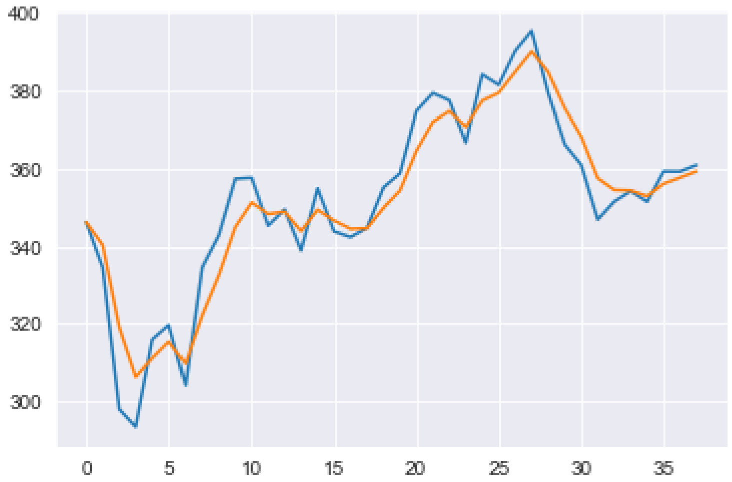
```
pandas.core.series.Series
```

In [78]:

```
# Try out the following with various values of 'alpha' and evaluate the results
ewma = dfo['Co2 Emission'].ewm(alpha=0.5, adjust=False).mean()
plt.plot(dfo['Co2 Emission'])
plt.plot(ewma)
```

Out[78]:

```
[<matplotlib.lines.Line2D at 0x205e0977f70>]
```

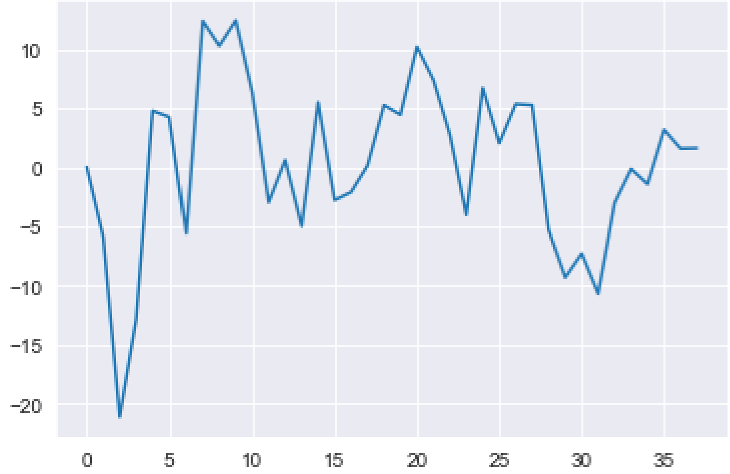


In [79]:

```
plt.plot(dfo['Co2 Emission']-ewma)
```

Out[79]:

```
[<matplotlib.lines.Line2D at 0x205e0a461a0>]
```



In [80]:

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

In [81]:

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

In [82]:

```
type(ses)
```

Out[82]:

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

In [83]:

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

In [84]:

```
result.summary()
```

Out[84]:

SimpleExpSmoothing Model Results

Dep. Variable:	Co2 Emission	**No. Observations:**	38
Model:	SimpleExpSmoothing	**SSE**	15706.520
Optimized:	False	**AIC**	232.921
Trend:	None	**BIC**	236.196
Seasonal:	None	**AICC**	234.133
Seasonal Periods:	None	**Date:**	Mon, 20 Feb 2023
Box-Cox:	False	**Time:**	12:06:34
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.1000000	alpha	False
initial_level	346.18372	l.0	False

In [85]:

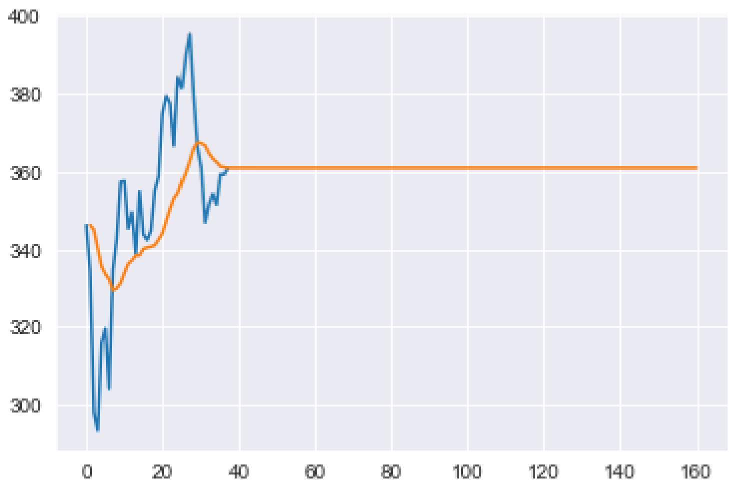
```
mypred = result.predict(start=1, end=160)
```

In [86]:

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

Out[86]:

```
[<matplotlib.lines.Line2D at 0x205e0adc940>]
```



In [87]:

```
result.params
```

Out[87]:

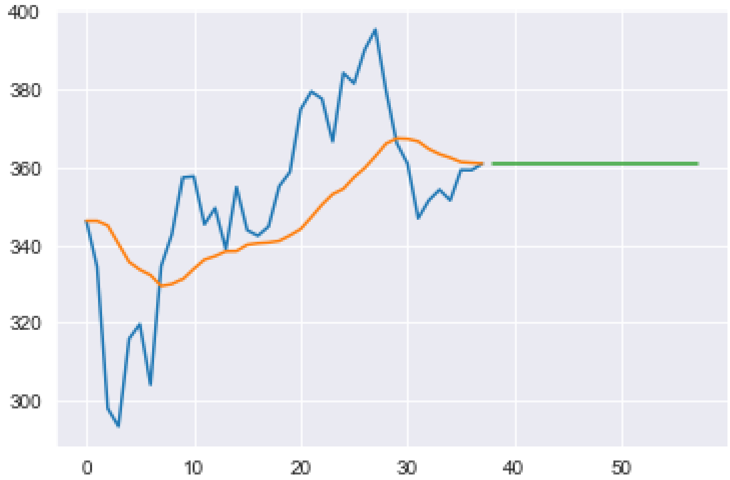
```
{'smoothing_level': 0.1,
 'smoothing_trend': None,
 'smoothing_seasonal': None,
 'damping_trend': nan,
 'initial_level': 346.183721,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [88]:

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

Out[88]:

```
[<matplotlib.lines.Line2D at 0x205e0b48fa0>]
```



In [89]:

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

In [90]:

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

Out[90]:

```
[<matplotlib.lines.Line2D at 0x205df7ba020>]
```

In [91]:

```
result2.params
```

Out[91]:

```
{'smoothing_level': 0.9999999850988388,
 'smoothing_trend': nan,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 346.18372671219953,
 'initial_trend': nan,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [92]:

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

In [93]:

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

Out[93]:

```
{'smoothing_level': 0.9999999850988388,
 'smoothing_trend': 0.0,
 'smoothing_seasonal': nan,
 'damping_trend': nan,
 'initial_level': 346.0103168251965,
 'initial_trend': 1.0005011818716572,
 'initial_seasons': array([], dtype=float64),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [94]:

```
result.summary()
```

Out[94]:

Holt Model Results

Dep. Variable:	Co2 Emission	**No. Observations:**	38
Model:	Holt	**SSE**	5803.630
Optimized:	True	**AIC**	199.089
Trend:	Multiplicative	**BIC**	205.639
Seasonal:	None	**AICC**	201.798
Seasonal Periods:	None	**Date:**	Mon, 20 Feb 2023
Box-Cox:	False	**Time:**	12:12:23
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	1.0000000	alpha	True
smoothing_trend	0.000000	beta	True
initial_level	346.01032	l.0	True
initial_trend	1.0005012	b.0	True

In [95]:

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

Out[95]:

```
[<matplotlib.lines.Line2D at 0x205e0bad990>]
```

In [96]:

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

In [97]:

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

In [98]:

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

Out[98]:

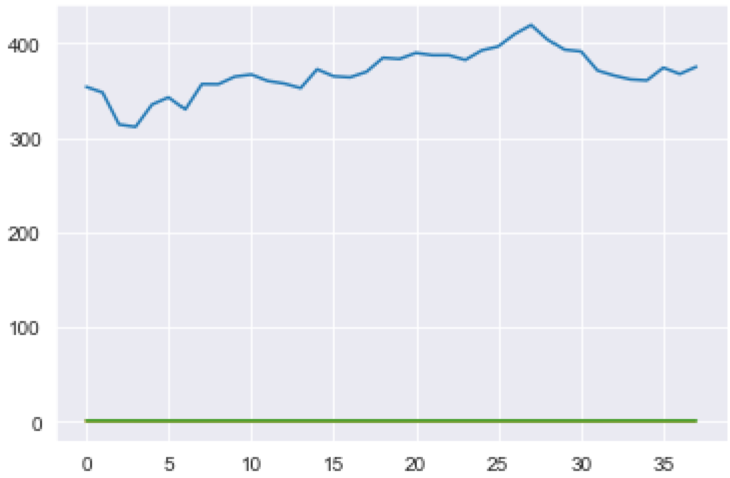
```
{'smoothing_level': 0.9745042450189672,
 'smoothing_trend': 4.006626951451572e-06,
 'smoothing_seasonal': 1.4492327099977186e-08,
 'damping_trend': nan,
 'initial_level': 353.3074976391492,
 'initial_trend': 1.0010387072000975,
 'initial_seasons': array([0.97831677, 0.96180932, 0.95180426, 0.942428
05, 0.94096637,
        0.93208535, 0.9225683 , 0.93606138, 0.96127516, 0.97965152,
        0.97472257, 0.95948718]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [99]:

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

Out[99]:

```
[<matplotlib.lines.Line2D at 0x205e0c14ca0>]
```



In [100]:

```
result3.summary()
```

Out[100]:

ExponentialSmoothing Model Results

Dep. Variable:	Co2 Emission	**No. Observations:**	38
Model:	ExponentialSmoothing	**SSE**	4781.401
Optimized:	True	**AIC**	215.726
Trend:	Multiplicative	**BIC**	241.928
Seasonal:	Multiplicative	**AICC**	251.726
Seasonal Periods:	12	**Date:**	Mon, 20 Feb 2023
Box-Cox:	False	**Time:**	12:13:36
Box-Cox Coeff.:	None		

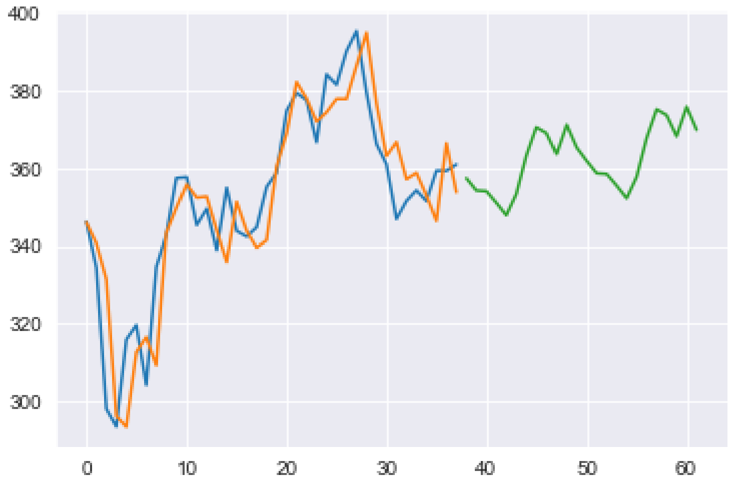
	coeff	code	optimized
smoothing_level	0.9745042	alpha	True
smoothing_trend	4.0066e-06	beta	True
smoothing_seasonal	1.4492e-08	gamma	True
initial_level	353.30750	l.0	True
initial_trend	1.0010387	b.0	True
initial_seasons.0	0.9783168	s.0	True
initial_seasons.1	0.9618093	s.1	True
initial_seasons.2	0.9518043	s.2	True
initial_seasons.3	0.9424281	s.3	True
initial_seasons.4	0.9409664	s.4	True
initial_seasons.5	0.9320853	s.5	True
initial_seasons.6	0.9225683	s.6	True
initial_seasons.7	0.9360614	s.7	True
initial_seasons.8	0.9612752	s.8	True
initial_seasons.9	0.9796515	s.9	True
initial_seasons.10	0.9747226	s.10	True
initial_seasons.11	0.9594872	s.11	True

In [101]:

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

Out[101]:

```
[<matplotlib.lines.Line2D at 0x205e0c6afe0>]
```



In [102]:

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

In [103]:

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

Out[103]:

```
{'smoothing_level': 0.9738882338534315,
 'smoothing_trend': 1.8594519555531137e-07,
 'smoothing_seasonal': 0.0,
 'damping_trend': nan,
 'initial_level': 320.7665101170618,
 'initial_trend': 0.5195289270219751,
 'initial_seasons': array([1.07690858, 1.05859404, 1.04759096, 1.037302
43, 1.03573098,
        1.02599132, 1.01555459, 1.03044311, 1.05819315, 1.07840055,
        1.07295981, 1.05619048]),
 'use_boxcox': False,
 'lamda': None,
 'remove_bias': False}
```

In [104]:

```
result4.summary()
```

Out[104]:

ExponentialSmoothing Model Results

Dep. Variable:	Co2 Emission	**No. Observations:**	38
Model:	ExponentialSmoothing	**SSE**	4775.081
Optimized:	True	**AIC**	215.676
Trend:	Additive	**BIC**	241.877
Seasonal:	Multiplicative	**AICC**	251.676
Seasonal Periods:	12	**Date:**	Mon, 20 Feb 2023
Box-Cox:	False	**Time:**	12:14:27
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.9738882	alpha	True
smoothing_trend	1.8595e-07	beta	True
smoothing_seasonal	0.000000	gamma	True
initial_level	320.76651	l.0	True
initial_trend	0.5195289	b.0	True
initial_seasons.0	1.0769086	s.0	True
initial_seasons.1	1.0585940	s.1	True
initial_seasons.2	1.0475910	s.2	True
initial_seasons.3	1.0373024	s.3	True
initial_seasons.4	1.0357310	s.4	True
initial_seasons.5	1.0259913	s.5	True
initial_seasons.6	1.0155546	s.6	True
initial_seasons.7	1.0304431	s.7	True
initial_seasons.8	1.0581932	s.8	True
initial_seasons.9	1.0784005	s.9	True
initial_seasons.10	1.0729598	s.10	True
initial_seasons.11	1.0561905	s.11	True

In [105]:

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

Out[105]:

```
[&lt;matplotlib.lines.Line2D at 0x205e0d01030&gt;]
```

In [ ]: