TimeSeries2021_Mar9_Forecasting

March 10, 2021

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

[3]:
```
df = pd.read_excel("indicator8_2014_1.xlsx")
```

[4]:
```
df.head()
```

[4]:
```
  Average Global Temperature, 1880-2013          Unnamed: 1  Unnamed: 2  \
0                                   NaN                 NaN         NaN
1                                  Year         Temperature         NaN
2                                   NaN  Degrees Fahrenheit         NaN
3                                   NaN                 NaN         NaN
4                                  1880              56.822         NaN

   Unnamed: 3  Unnamed: 4
0         NaN         NaN
1         NaN         NaN
2         NaN         NaN
3         NaN         NaN
4         NaN         NaN
```

[5]:
```
df.tail()
```

[5]:
```
                 Average Global Temperature, 1880-2013 Unnamed: 1  Unnamed: 2  \
136                                               2012     58.244         NaN
137                                               2013     58.298         NaN
138                                                NaN        NaN         NaN
139  Source: Compiled by Earth Policy Institute fro…        NaN         NaN
140                                                NaN        NaN         NaN

     Unnamed: 3  Unnamed: 4
136         NaN         NaN
137         NaN         NaN
138         NaN         NaN
139         NaN         NaN
```

```
140          NaN          NaN
```

[6]:
```
# Add parameters to fix various problems.
df = pd.read_excel("indicator8_2014_1.xlsx", index_col=0, usecols="A:B",
                   skiprows=4, skipfooter=3,
                   parse_dates=True, names=["Date","Temp"])
```

[7]:
```
df.head()
```

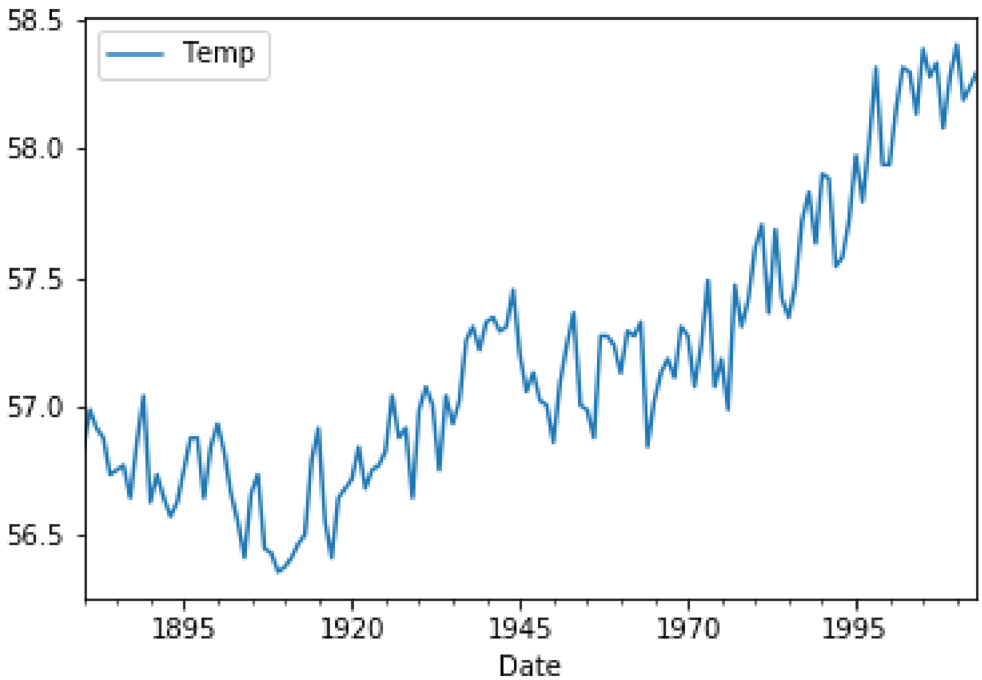
[7]:
```
              Temp
Date
1880-01-01  56.822
1881-01-01  56.984
1882-01-01  56.912
1883-01-01  56.876
1884-01-01  56.732
```

[8]:
```
df.tail()
```

[8]:
```
              Temp
Date
2009-01-01  58.280
2010-01-01  58.406
2011-01-01  58.190
2012-01-01  58.244
2013-01-01  58.298
```

[9]:
```
df.plot()
```

[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11d0f8f10>
```



[10]:
```
type(df.index[0])
```

[10]: pandas._libs.tslibs.timestamps.Timestamp

Rather than treat the average temperature over a particular year has happening at a particular time stamp, instead we could consider the average temperature to be happening over a particular ***time period***. Python can distinguish between the two.

[11]:
```
df.index = df.index.to_period('Y')
```

[13]:
```
type(df.index[0])
```

[13]: pandas._libs.tslibs.period.Period

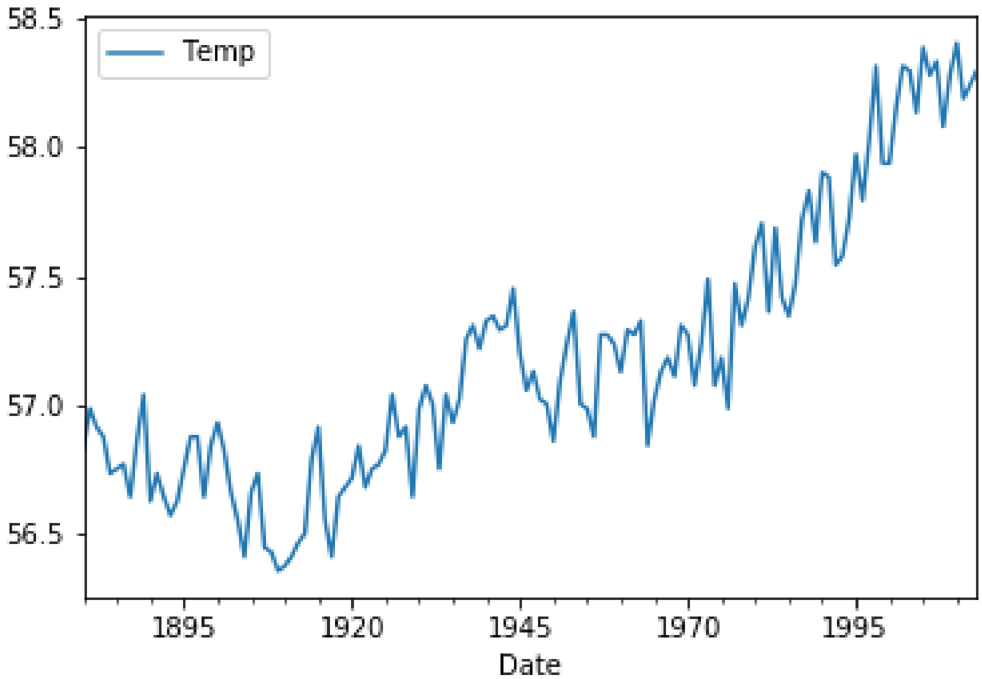
[14]:
```
df.head()
```

[14]:

Date	Temp
1880	56.822
1881	56.984
1882	56.912
1883	56.876
1884	56.732

[15]:
```
df.plot()
```

[15]: <matplotlib.axes._subplots.AxesSubplot at 0x111a49e10>



[16]:
```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

[17]:
```
model = ExponentialSmoothing(df)
```

[18]:
```
# This is simple exponential smoothing.
fit = model.fit()
fit.summary()
```

[18]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                       ExponentialSmoothing Model Results
================================================================================
Dep. Variable:                    endog   No. Observations:                  134
Model:             ExponentialSmoothing   SSE                              3.828
Optimized:                         True   AIC                           -472.428
Trend:                             None   BIC                           -466.632
Seasonal:                          None   AICC                          -472.118
Seasonal Periods:                  None   Date:                 Wed, 10 Mar 2021
Box-Cox:                          False   Time:                         23:15:09
Box-Cox Coeff.:                    None
==============================================================================
                       coeff                 code              optimized
```

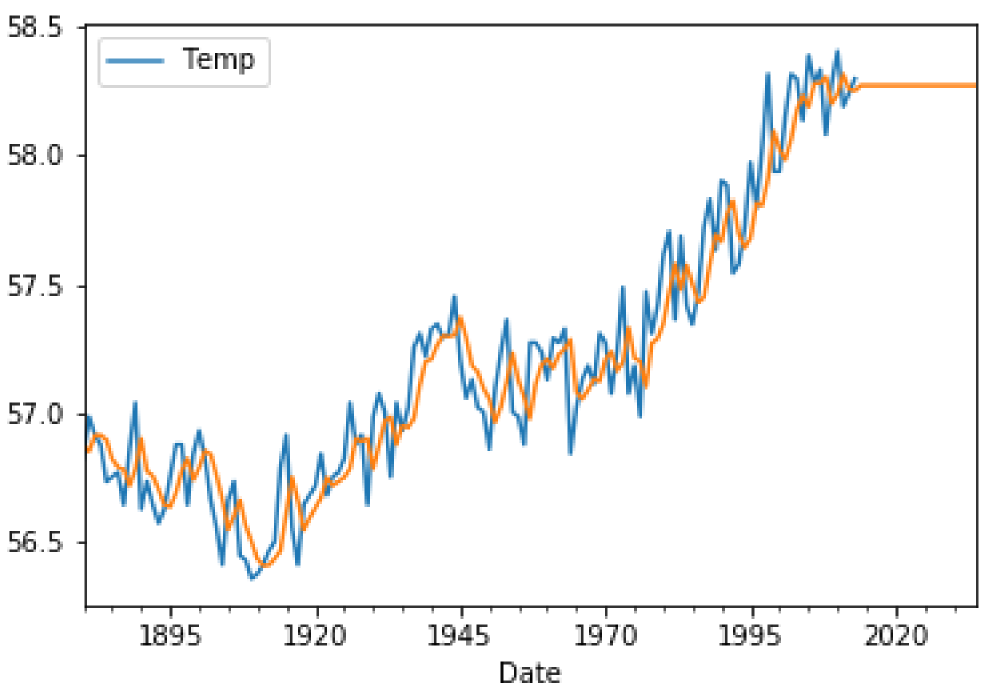
```
==============================================================================
smoothing_level            0.4601695                alpha                 True
initial_level              56.871623                  1.0                 True
------------------------------------------------------------------------------
"""
```

[19]:
```
dfhat = fit.predict(0,len(df)+20)
```

[20]:
```
plt.figure()
df.plot()
dfhat.plot()
```

[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1261986d0>
```

```
<Figure size 432x288 with 0 Axes>
```



[21]:
```
# This is Holt with trend (also called double exponential smoothing).
# But here the best fit turned out to be a constant trend.
model = ExponentialSmoothing(df, trend="additive")
fit = model.fit()
fit.summary()
```

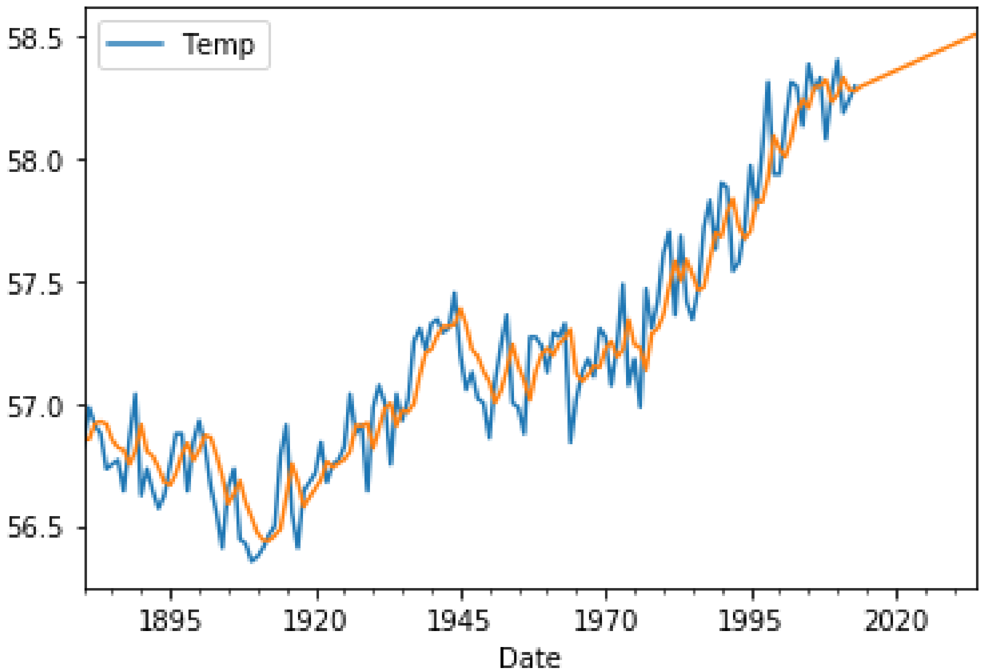
[21]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

```
                     ExponentialSmoothing Model Results
================================================================================
Dep. Variable:                    endog   No. Observations:                  134
Model:             ExponentialSmoothing   SSE                              3.750
Optimized:                         True   AIC                           -471.195
Trend:                         Additive   BIC                           -459.604
Seasonal:                          None   AICC                          -470.534
Seasonal Periods:                  None   Date:                 Wed, 10 Mar 2021
Box-Cox:                          False   Time:                         23:15:15
Box-Cox Coeff.:                    None
==============================================================================
                       coeff                 code              optimized
------------------------------------------------------------------------------
smoothing_level            0.4156884                alpha                 True
smoothing_slope             0.000000                 beta                 True
initial_level              56.843829                  l.0                 True
initial_slope              0.0107676                  b.0                 True
------------------------------------------------------------------------------
"""
```

[22]:
```
dfhat = fit.predict(0,len(df)+20)
plt.figure()
df.plot()
dfhat.plot()
```

[22]: <matplotlib.axes._subplots.AxesSubplot at 0x12614e650>

<Figure size 432x288 with 0 Axes>



[23]:
```
# The "Pyramid ARIMA" package has a R-style "auto.arima" function.  To install:
# pip install pmdarima
import pmdarima as pm
```

[24]:
```
fit = pm.auto_arima(df)  # auto_arima returns a fitted model
```

[25]:
```
fit.summary()
```

[25]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  134
Model:               SARIMAX(1, 1, 3)   Log Likelihood                  53.392
Date:                Wed, 10 Mar 2021   AIC                            -94.784
Time:                        23:15:32   BIC                            -77.441
Sample:                             0   HQIC                           -87.736
                                - 134
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0216      0.010      2.129      0.033       0.002       0.041
ar.L1         -0.9478      0.063    -14.969      0.000      -1.072      -0.824
ma.L1          0.5644      0.112      5.054      0.000       0.346       0.783
```

```
ma.L2         -0.5824      0.095     -6.154      0.000      -0.768      -0.397
ma.L3         -0.2821      0.091     -3.101      0.002      -0.460      -0.104
sigma2         0.0261      0.004      6.888      0.000       0.019       0.034
================================================================================
===
Ljung-Box (Q):                       39.46   Jarque-Bera (JB):
1.79
Prob(Q):                              0.49   Prob(JB):
0.41
Heteroskedasticity (H):               1.55   Skew:
-0.09
Prob(H) (two-sided):                  0.15   Kurtosis:
2.46
================================================================================
===

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

[26]:

```
forecasts = pd.Series(fit.predict(100), index=pd.period_range('2014',␣
↪periods=100, freq='Y'))
```

[27]:

```
forecasts.head()
```

[27]:

```
2014    58.287975
2015    58.319309
2016    58.317231
2017    58.340785
2018    58.340045
Freq: A-DEC, dtype: float64
```

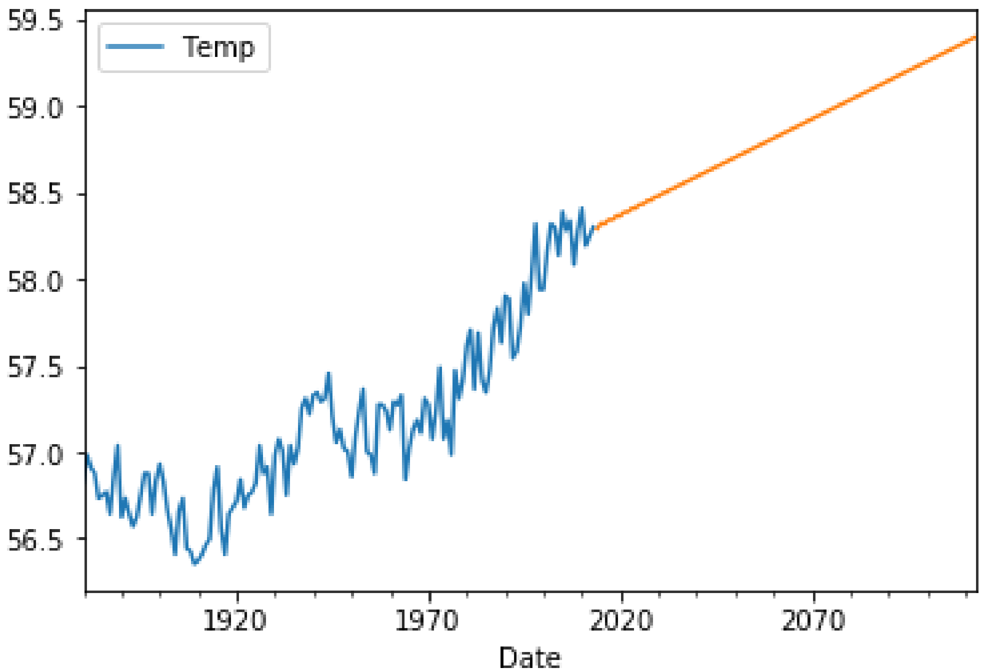
[28]:

```
plt.figure()
df.plot()
forecasts.plot()
```

[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12a0d3050>
```

```
<Figure size 432x288 with 0 Axes>
```



[29]:
```
# We can also use "predict" to return confidence intervals.
fit.predict(100, return_conf_int=True)
```

[29]:
```
(array([58.28797495, 58.31930892, 58.31723148, 58.34078499, 58.34004505,
        58.36233085, 58.36279247, 58.38393942, 58.38548047, 58.40560432,
        58.40811507, 58.42731981, 58.43070171, 58.44908077, 58.45324527,
        58.47088256, 58.47575011, 58.49272105, 58.49822019, 58.51459249,
        58.52065903, 58.53649354, 58.5430698 , 58.55842119, 58.56545536,
        58.58037273, 58.58781827, 58.60234574, 58.61016084, 58.62433803,
        58.63248513, 58.64634765, 58.654793  , 58.66837284, 58.67708611,
        58.690412  , 58.69936598, 58.71246373, 58.72163394, 58.73452673,
        58.7438912 , 58.75659988, 58.76613886, 58.77868213, 58.78837789,
        58.80077257, 58.81060916, 58.82287035, 58.83283346, 58.84497473,
        58.85505151, 58.86708505, 58.87726394, 58.88920069, 58.89947131,
        58.91132113, 58.92167415, 58.93344586, 58.94387291, 58.95557445,
        58.96606802, 58.97770652, 58.98825983, 58.9998417 , 59.01044868,
        59.02197968, 59.03263489, 59.04412018, 59.0548187 , 59.06626295,
        59.07700038, 59.08840774, 59.09918013, 59.11055435, 59.12135815,
        59.13270261, 59.14353462, 59.15485233, 59.16570969, 59.17700338,
        59.18788351, 59.19915562, 59.2100562 , 59.22130892, 59.23222788,
        59.24346319, 59.25439865, 59.26561831, 59.27656861, 59.28777421,
        59.29873783, 59.30993081, 59.32090639, 59.33208803, 59.34307436,
        59.35424581, 59.3652418 , 59.37640409, 59.38740876, 59.39856283]),
 array([[57.9713109 , 58.604639  ],
```

```
[57.94728736, 58.69133047],
[57.92448469, 58.70997826],
[57.93493391, 58.74663606],
[57.91547875, 58.76461136],
[57.92533913, 58.79932258],
[57.90866846, 58.81691649],
[57.91794381, 58.84993503],
[57.90363592, 58.86732501],
[57.91235465, 58.89885399],
[57.90008019, 58.91614995],
[57.90828253, 58.9463571 ],
[57.89777642, 58.96362701],
[57.90550756, 58.99265398],
[57.8965516 , 59.00993894],
[57.90385787, 59.03790726],
[57.89626941, 59.05523082],
[57.90319613, 59.08224597],
[57.89682026, 59.09962012],
[57.90341055, 59.12577443],
[57.89811454, 59.14320351],
[57.90440872, 59.16857836],
[57.9000779 , 59.1860617 ],
[57.90611321, 59.21072916],
[57.90264785, 59.22826287],
[57.90845841, 59.25228705],
[57.90577128, 59.26986526],
[57.91138811, 59.29330336],
[57.90940262, 59.31091907],
[57.9148538 , 59.33382227],
[57.91350236, 59.3514679 ],
[57.9188132 , 59.3738821 ],
[57.91803603, 59.39154996],
[57.92322927, 59.4135164 ],
[57.92297331, 59.43119891],
[57.92806929, 59.45275471],
[57.92828736, 59.4704446 ],
[57.93330424, 59.49162322],
[57.93395425, 59.50931363],
[57.93890821, 59.53014526],
[57.93995257, 59.54782984],
[57.94485798, 59.56834178],
[57.94626304, 59.58601468],
[57.95113264, 59.60623163],
[57.95286823, 59.62388754],
[57.95771328, 59.64383186],
[57.95975232, 59.661466  ],
[57.96458273, 59.68115796],
```

```
[57.96690087, 59.69876606],
[57.97172538, 59.71822408],
[57.97430067, 59.73580235],
[57.97912694, 59.75504315],
[57.98193961, 59.77258826],
[57.98677434, 59.79162705],
[57.98980652, 59.8091361 ],
[57.99465553, 59.82798672],
[57.99789109, 59.84545721],
[58.00275945, 59.86413227],
[58.00618378, 59.88156205],
[58.01107586, 59.90007305],
[58.01467571, 59.91746032],
[58.01959528, 59.93581775],
[58.02335865, 59.95316101],
[58.02830893, 59.97137447],
[58.0322249 , 59.98867247],
[58.03720863, 60.00675074],
[58.04126727, 60.0240025 ],
[58.04628676, 60.04195361],
[58.05047904, 60.05915837],
[58.05553622, 60.07698967],
[58.0598539 , 60.09414686],
[58.06495035, 60.11186513],
[58.06938591, 60.12897434],
[58.07452292, 60.14658578],
[58.0790695 , 60.1636468 ],
[58.0842481 , 60.18115711],
[58.08889941, 60.19816982],
[58.0941204 , 60.21558426],
[58.09887067, 60.2325487 ],
[58.10413465, 60.24987212],
[58.1089786 , 60.26678841],
[58.11428598, 60.28402526],
[58.11921876, 60.30089364],
[58.12456979, 60.31804806],
[58.12958693, 60.33486883],
[58.13498174, 60.35194463],
[58.14007914, 60.36871817],
[58.14551774, 60.38571889],
[58.15069159, 60.40244563],
[58.15617388, 60.41937455],
[58.16142069, 60.43605497],
[58.16694647, 60.45291514],
[58.172263  , 60.46954979],
[58.17783202, 60.48634403],
[58.18321527, 60.50293346],
```

```
       [58.18882719, 60.51966442],
       [58.19427438, 60.53620923],
       [58.19992883, 60.55287935],
       [58.20543736, 60.56938017],
       [58.2111339 , 60.58599175]]))
```

[30]:
```
forecasts = pd.DataFrame(fit.predict(100, return_conf_int=True)[1],
                        index=pd.period_range('2014', periods=100, freq='Y'))
```

[31]:
```
forecasts.head()
```

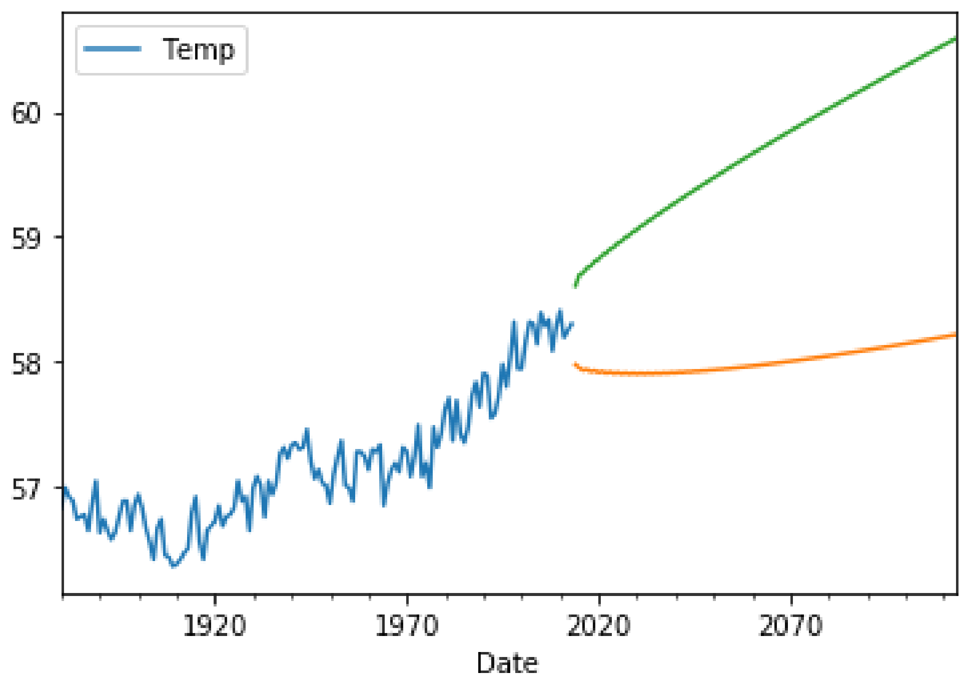
[31]:

	0	1
2014	57.971311	58.604639
2015	57.947287	58.691330
2016	57.924485	58.709978
2017	57.934934	58.746636
2018	57.915479	58.764611

[32]:
```
plt.figure()
df.plot()
forecasts[0].plot()
forecasts[1].plot()
```

[32]: <matplotlib.axes._subplots.AxesSubplot at 0x12a0845d0>

<Figure size 432x288 with 0 Axes>



[33]:
```python
# Let's fit a trend stationary rather than difference stationary model.
fit = pm.auto_arima(df, d=0, suppress_warnings=True, trend="ct")
fit.summary()
```

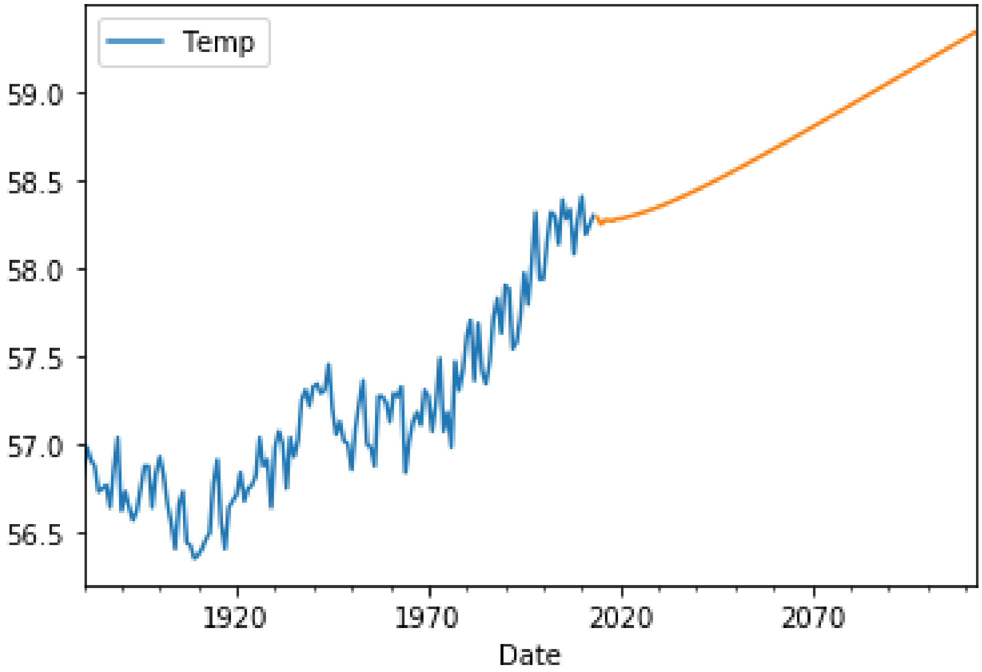
[33]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  134
Model:               SARIMAX(3, 0, 3)   Log Likelihood                  53.958
Date:                Wed, 10 Mar 2021   AIC                            -89.916
Time:                        23:17:28   BIC                            -63.836
Sample:                             0   HQIC                           -79.318
                                - 134
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      7.8362      6.228      1.258      0.208      -4.370      20.042
drift          0.0018      0.001      1.512      0.131      -0.001       0.004
ar.L1          0.1448      0.477      0.303      0.762      -0.791       1.080
ar.L2          0.3471      0.519      0.669      0.504      -0.670       1.365
ar.L3          0.3695      0.332      1.114      0.265      -0.281       1.020
ma.L1          0.3589      0.469      0.765      0.444      -0.560       1.278
ma.L2         -0.1600      0.406     -0.395      0.693      -0.955       0.635
ma.L3         -0.3735      0.185     -2.020      0.043      -0.736      -0.011
sigma2         0.0261      0.004      6.699      0.000       0.018       0.034
================================================================================
===
Ljung-Box (Q):                       41.37   Jarque-Bera (JB):
2.55
Prob(Q):                              0.41   Prob(JB):
0.28
Heteroskedasticity (H):               1.42   Skew:
-0.26
Prob(H) (two-sided):                  0.24   Kurtosis:
2.56
================================================================================
===

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

[34]:
```
forecasts = pd.Series(fit.predict(100), index=pd.period_range('2014',␣
↪periods=100, freq='Y'))
plt.figure()
df.plot()
forecasts.plot()
```

[34]: <matplotlib.axes._subplots.AxesSubplot at 0x12a1ae910>

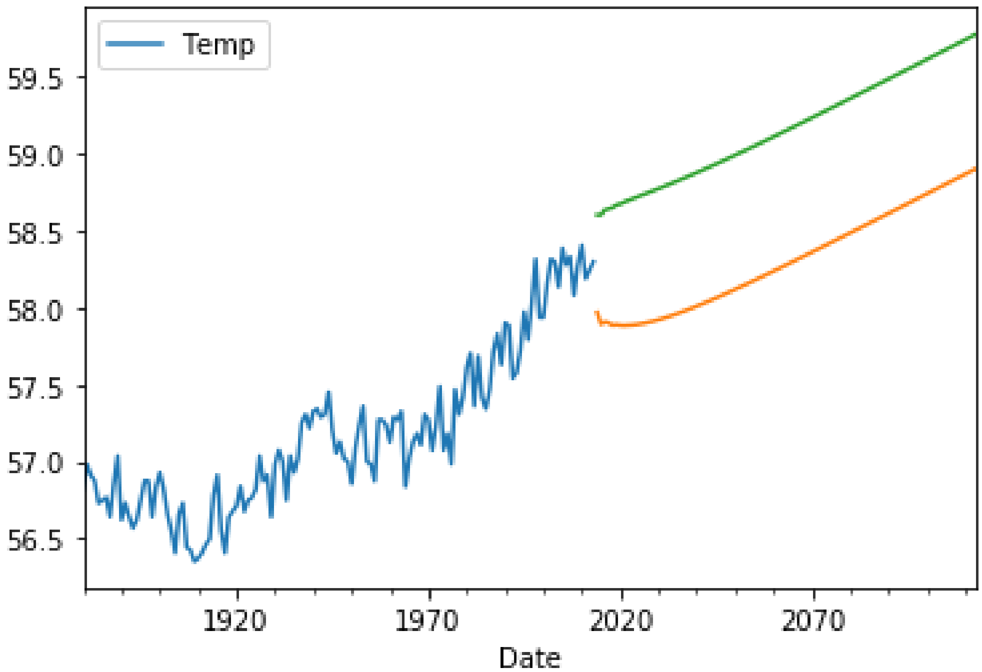
<Figure size 432x288 with 0 Axes>



[35]:
```
forecasts = pd.DataFrame(fit.predict(100, return_conf_int=True)[1],
                         index=pd.period_range('2014', periods=100, freq='Y'))
plt.figure()
df.plot()
forecasts[0].plot()
forecasts[1].plot()
```

[35]: <matplotlib.axes._subplots.AxesSubplot at 0x12a3454d0>

<Figure size 432x288 with 0 Axes>



Note that forecast confidence intervals for the trend stationary model don't get bigger (in contrast with the difference stationary model).

[ ]: