In [1]:
```
import pandas as pd
import datetime
import regex as re
import math
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
from itertools import cycle
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error, explained
from sklearn.metrics import mean_poisson_deviance, mean_gamma_deviance, accura
from sklearn.preprocessing import MinMaxScaler

import warnings
warnings.filterwarnings('ignore')

FILE_NAME = "./2023-06-01-13-18-37.csv"

eth = pd.read_csv(FILE_NAME)

print(f"{len(eth)} rows")

eth["Date"] = pd.to_datetime(eth['Date'])

last_date = eth["Date"].max()

print(f"Latest row is from {last_date}")
```

```
2555 rows
Latest row is from 2023-03-08 00:00:00
```

Out[1]:

	Date	Price	Open	High	Low	Vol	Change
0	2023-03-08	1553.49	1561.79	1569.70	1548.98	498570	-0.53
1	2023-03-07	1561.78	1565.84	1580.95	1536.31	460100	-0.26
2	2023-03-06	1565.84	1564.36	1581.13	1555.43	322160	0.09
3	2023-03-05	1564.37	1566.73	1587.95	1556.84	313010	-0.15
4	2023-03-04	1566.73	1569.45	1577.02	1550.10	247020	-0.14

In [2]:
```
eth.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2555 entries, 0 to 2554
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   Date    2555 non-null   datetime64[ns]
 1   Price   2555 non-null   float64       
 2   Open    2555 non-null   float64       
 3   High    2555 non-null   float64       
 4   Low     2555 non-null   float64       
 5   Vol     2555 non-null   object        
 6   Change  2555 non-null   object        
dtypes: datetime64[ns](1), float64(4), object(2)
memory usage: 139.9+ KB
```

In [3]:
```
eth.describe()
```

Out[3]:
```
(2555, 7)
```

In [4]:
```
print('Total number of days :', eth.Date.nunique())
print('Total number of fields :', eth.shape[1])
```

```
Total number of days : 2555
Total number of fields : 7
```

In [5]:
```
print("Null values :", eth.isnull().values.sum())
print("NA values :", eth.isna().values.any())
```

```
Null values : 0
NA values : False
```

In [6]:
```
print("Starting date :", eth.iloc[-1][0])
print("Ending date :", eth.iloc[0][0])
print("Duration :", eth.iloc[0][0] - eth.iloc[-1][0])
```

```
Starting date : 2016-03-10 00:00:00
Ending date : 2023-03-08 00:00:00
Duration : 2554 days 00:00:00
```

In [7]:
```
monthwise = eth.groupby(pd.DatetimeIndex(eth.Date).month)[['Open']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', '
             'September', 'October', 'November', 'December']
monthwise = monthwise.reset_index()
monthwise['Date'] = new_order
```

Out[7]:

	Date	Open
0	January	1012.926636
1	February	1057.254670
2	March	856.974306
3	April	897.661762
4	May	940.999447
5	June	729.158619
6	July	666.152673
7	August	857.359770
8	September	848.079286
9	October	888.357926
10	November	989.121476
11	December	971.279631

In [8]:
```
fig = go.Figure()

fig.add_trace(go.Bar(
    x = monthwise.Date,
    y = monthwise['Open'],
    name = 'Stock Open Price',
    marker_color = 'black'
))
fig.update_layout(barmode = 'group', xaxis_tickangle = -45,
                  title = 'Monthwise comparision for Open Prices')
```

In [9]:
```
monthwise_high = eth.groupby(pd.DatetimeIndex(eth.Date).month)['High'].max()
monthwise_high = monthwise_high.reset_index()
monthwise_high['Date'] = new_order

monthwise_low = eth.groupby(pd.DatetimeIndex(eth.Date).month)['Low'].min()
monthwise_low = monthwise_low.reset_index()
```

In [10]:
```
fig = go.Figure()
fig.add_trace(go.Bar(
    x = monthwise_high.Date,
    y = monthwise_high.High,
    name = 'Stock High Price',
    marker_color = 'purple'
))
fig.add_trace(go.Bar(
    x = monthwise_low.Date,
    y = monthwise_low.Low,
    name = 'Stock Low Price',
    marker_color='pink'
))

fig.update_layout(barmode='group', xaxis_tickangle = -45,
                  title=' Monthwise High and Low Price')
```

In [11]:
```
names = cycle(['Eth Open Price','Eth High Price','Eth Low Price'])

fig = px.line(eth, x = eth.Date, y = [eth['Open'], eth['High'], eth['Low']],
              labels = {'date': 'Date','value':'Eth value'})
fig.update_layout(title_text = 'Ethereum Price analysis chart', font_size = 15
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid = False)
fig.update_yaxes(showgrid = False)

fig.show()
```

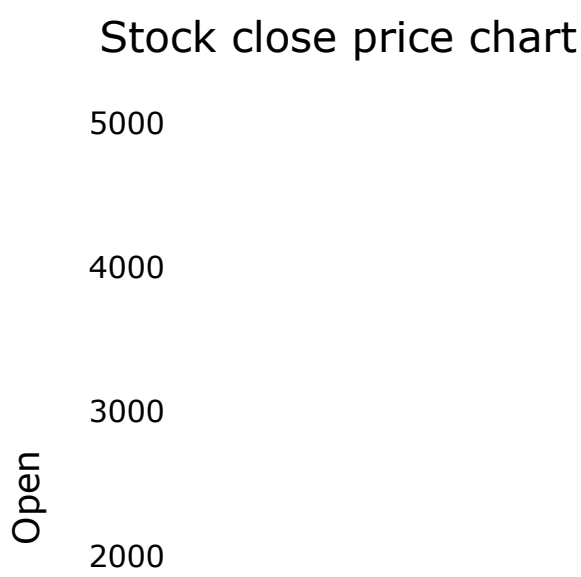
In [12]:
```
open_eth = eth[['Date', 'Open']]
print(open_eth.shape)
```

```
(2555, 2)
```

Out[12]:

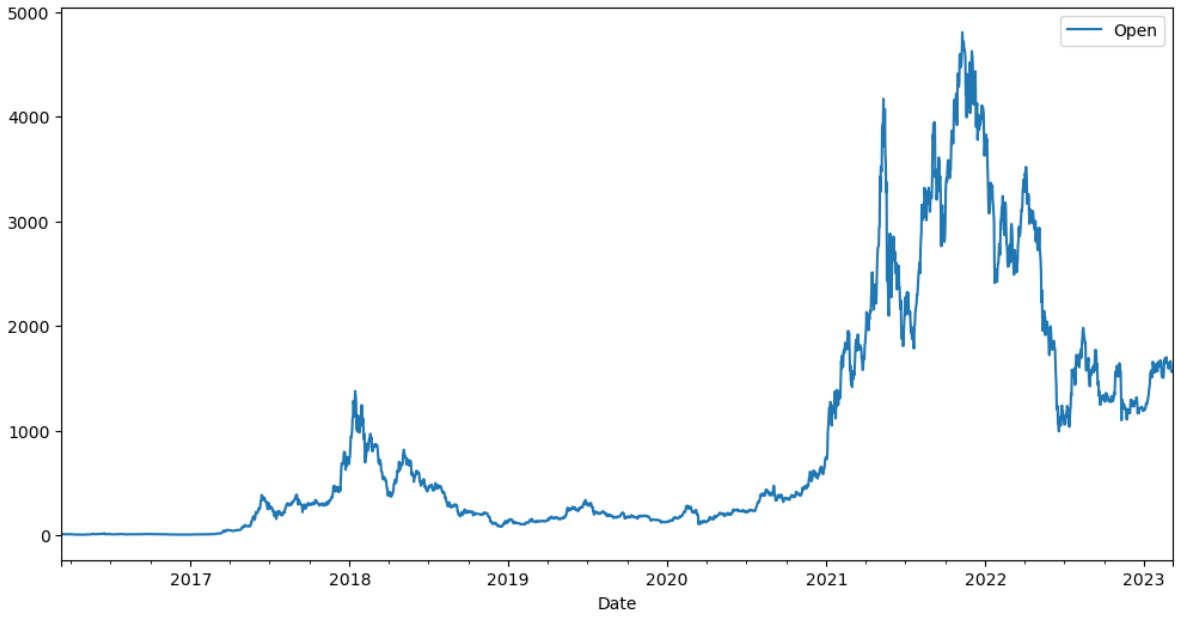
	Date	Open
0	2023-03-08	1561.79
1	2023-03-07	1565.84
2	2023-03-06	1564.36
3	2023-03-05	1566.73
4	2023-03-04	1569.45

In [13]:
```
fig = px.line(open_eth, x = open_eth.Date, y = open_eth.Open,labels = {'date':
fig.update_traces(marker_line_width = 2, opacity = 0.8)
fig.update_layout(title_text = 'Stock close price chart', plot_bgcolor = 'whit
fig.update_xaxes(showgrid = False)
fig.update_yaxes(showgrid = False)
```



In [14]:

Out[14]: <AxesSubplot: xlabel='Date'>



In [15]:
```
prophet_data = eth[["Date", "Open"]]

prophet_data = prophet_data.rename(columns = {
    "Date": "ds",
    "Open": "y"
})
```

Out[15]:

	ds	y
0	2023-03-08	1561.79
1	2023-03-07	1565.84
2	2023-03-06	1564.36
3	2023-03-05	1566.73
4	2023-03-04	1569.45

In [16]:

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) http
s://pip.repos.neuron.amazonaws.com (https://pip.repos.neuron.amazonaws.com)
Requirement already satisfied: prophet in /home/ec2-user/anaconda3/envs/pytho
n3/lib/python3.10/site-packages (1.1.4)
Requirement already satisfied: LunarCalendar>=0.0.9 in /home/ec2-user/anacond
a3/envs/python3/lib/python3.10/site-packages (from prophet) (0.0.9)
Requirement already satisfied: convertdate>=2.1.2 in /home/ec2-user/anaconda3
/envs/python3/lib/python3.10/site-packages (from prophet) (2.4.0)
Requirement already satisfied: pandas>=1.0.4 in /home/ec2-user/anaconda3/envs
/python3/lib/python3.10/site-packages (from prophet) (1.5.2)
Requirement already satisfied: holidays>=0.25 in /home/ec2-user/anaconda3/env
s/python3/lib/python3.10/site-packages (from prophet) (0.25)
Requirement already satisfied: python-dateutil>=2.8.0 in /home/ec2-user/anaco
nda3/envs/python3/lib/python3.10/site-packages (from prophet) (2.8.2)
Requirement already satisfied: importlib-resources in /home/ec2-user/anaconda
3/envs/python3/lib/python3.10/site-packages (from prophet) (5.10.2)
Requirement already satisfied: matplotlib>=2.0.0 in /home/ec2-user/anaconda3/
envs/python3/lib/python3.10/site-packages (from prophet) (3.6.2)
Requirement already satisfied: numpy>=1.15.4 in /home/ec2-user/anaconda3/envs
/python3/lib/python3.10/site-packages (from prophet) (1.22.3)
Requirement already satisfied: cmdstanpy>=1.0.4 in /home/ec2-user/anaconda3/e
nvs/python3/lib/python3.10/site-packages (from prophet) (1.1.0)
Requirement already satisfied: tqdm>=4.36.1 in /home/ec2-user/anaconda3/envs/
python3/lib/python3.10/site-packages (from prophet) (4.64.1)
Requirement already satisfied: pymeeus<=1,>=0.3.13 in /home/ec2-user/anaconda
3/envs/python3/lib/python3.10/site-packages (from convertdate>=2.1.2->prophe
t) (0.5.12)
Requirement already satisfied: korean-lunar-calendar in /home/ec2-user/anacon
da3/envs/python3/lib/python3.10/site-packages (from holidays>=0.25->prophet)
(0.3.1)
Requirement already satisfied: pytz in /home/ec2-user/anaconda3/envs/python3/
lib/python3.10/site-packages (from LunarCalendar>=0.0.9->prophet) (2022.7)
Requirement already satisfied: ephem>=3.7.5.3 in /home/ec2-user/anaconda3/env
s/python3/lib/python3.10/site-packages (from LunarCalendar>=0.0.9->prophet)
(4.1.4)
Requirement already satisfied: pillow>=6.2.0 in /home/ec2-user/anaconda3/envs
/python3/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet) (9.4.
0)
Requirement already satisfied: pyparsing>=2.2.1 in /home/ec2-user/anaconda3/e
nvs/python3/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet)
(3.0.9)
Requirement already satisfied: contourpy>=1.0.1 in /home/ec2-user/anaconda3/e
nvs/python3/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet)
(1.0.6)
Requirement already satisfied: cycler>=0.10 in /home/ec2-user/anaconda3/envs/
python3/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet) (0.11.
0)
Requirement already satisfied: fonttools>=4.22.0 in /home/ec2-user/anaconda3/
envs/python3/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet)
(4.38.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /home/ec2-user/anaconda3/
envs/python3/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet)
(1.4.4)
Requirement already satisfied: packaging>=20.0 in /home/ec2-user/anaconda3/en
```

```
vs/python3/lib/python3.10/site-packages (from matplotlib>=2.0.0->prophet) (2
1.3)
Requirement already satisfied: six>=1.5 in /home/ec2-user/anaconda3/envs/pyth
on3/lib/python3.10/site-packages (from python-dateutil>=2.8.0->prophet) (1.1
6.0)
Note: you may need to restart the kernel to use updated packages.
```

In [17]:
```
from prophet import Prophet

prophet = Prophet(daily_seasonality=True)

prophet.fit(prophet_data)
```

```
10:16:57 - cmdstanpy - INFO - Chain [1] start processing
10:16:58 - cmdstanpy - INFO - Chain [1] done processing

Data fitted
```

In [18]:
```
future = prophet.make_future_dataframe(periods=365, include_history=False)
```

Out[18]:

	ds
360	2024-03-03
361	2024-03-04
362	2024-03-05
363	2024-03-06
364	2024-03-07

In [19]:
```
forecast = prophet.predict(future)
```

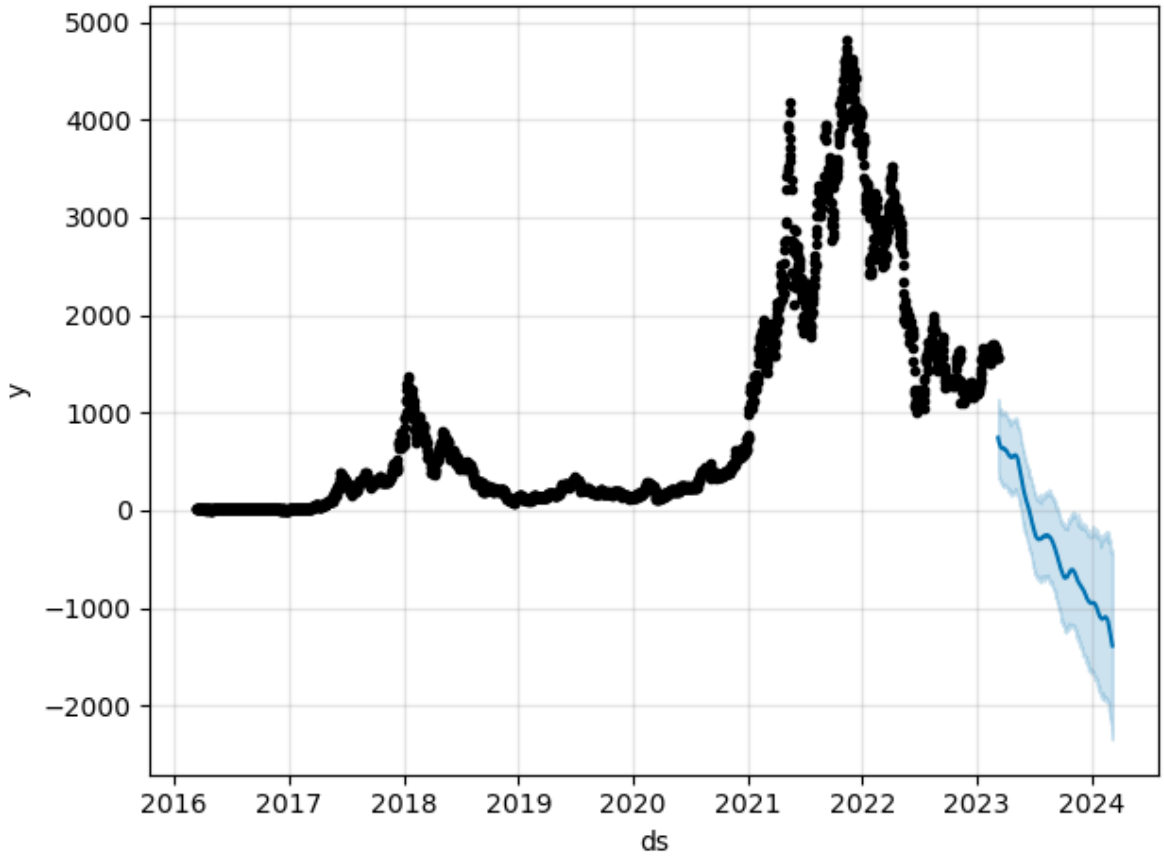
Out[19]:

	ds	yhat	yhat_lower	yhat_upper
360	2024-03-03	-1313.602477	-2191.856692	-406.585713
361	2024-03-04	-1332.073425	-2278.967714	-418.061722
362	2024-03-05	-1351.156908	-2277.478256	-441.452185
363	2024-03-06	-1369.540494	-2299.065918	-446.676894
364	2024-03-07	-1386.660811	-2347.995045	-401.149564

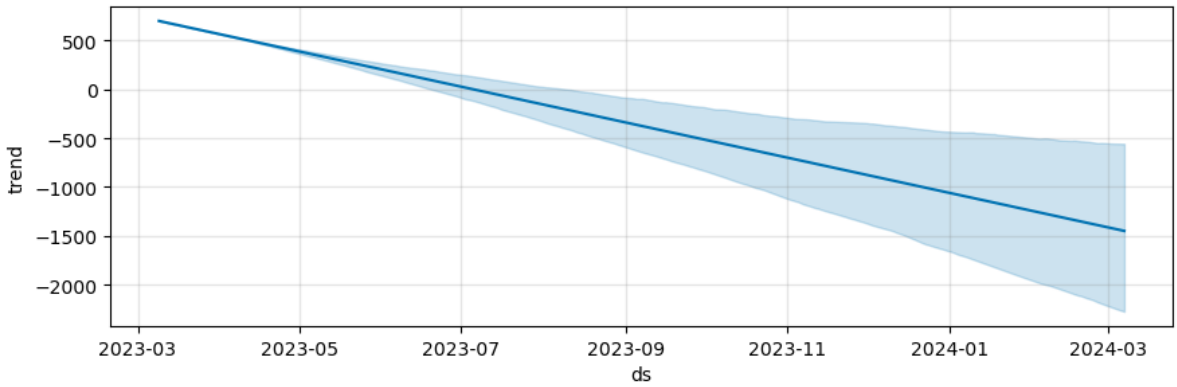
In [20]:

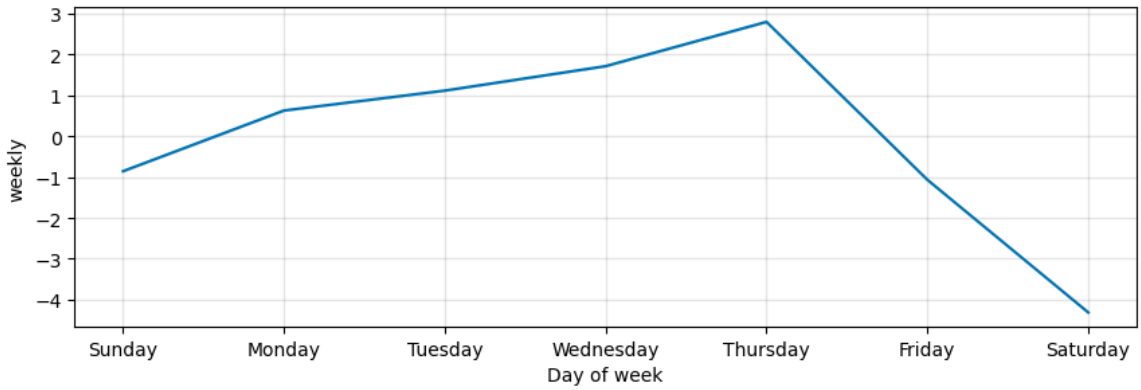
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 22 columns):
 #   Column                      Non-Null Count  Dtype         
---  ------                      --------------  -----         
 0   ds                          365 non-null    datetime64[ns]
 1   trend                       365 non-null    float64       
 2   yhat_lower                  365 non-null    float64       
 3   yhat_upper                  365 non-null    float64       
 4   trend_lower                 365 non-null    float64       
 5   trend_upper                 365 non-null    float64       
 6   additive_terms              365 non-null    float64       
 7   additive_terms_lower        365 non-null    float64       
 8   additive_terms_upper        365 non-null    float64       
 9   daily                       365 non-null    float64       
 10  daily_lower                 365 non-null    float64       
 11  daily_upper                 365 non-null    float64       
 12  weekly                      365 non-null    float64       
 13  weekly_lower                365 non-null    float64       
 14  weekly_upper                365 non-null    float64       
 15  yearly                      365 non-null    float64       
 16  yearly_lower                365 non-null    float64       
 17  yearly_upper                365 non-null    float64       
 18  multiplicative_terms        365 non-null    float64       
 19  multiplicative_terms_lower  365 non-null    float64       
 20  multiplicative_terms_upper  365 non-null    float64       
 21  yhat                        365 non-null    float64       
dtypes: datetime64[ns](1), float64(21)
memory usage: 62.9 KB
```

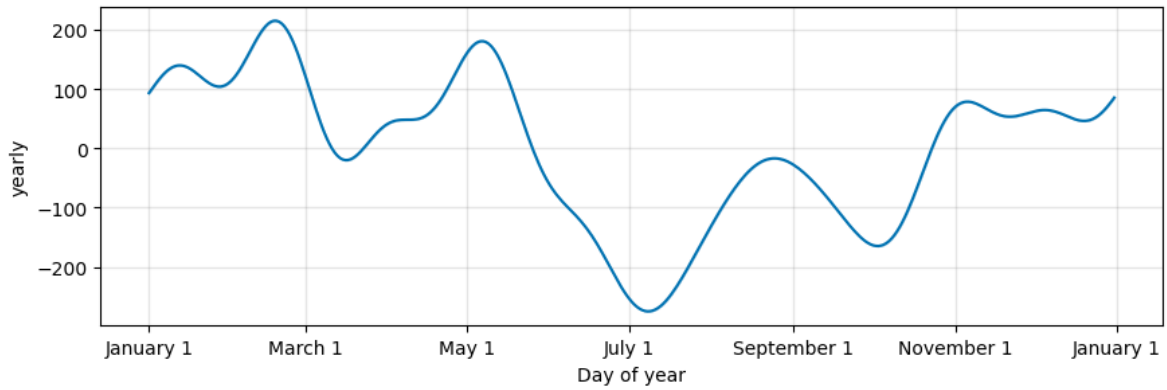
In [21]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt

fig = plt.figure(dpi=100)

fig.set_facecolor("white")

prophet_plot_forecast_fig = prophet.plot(forecast, ax=fig.gca());
```

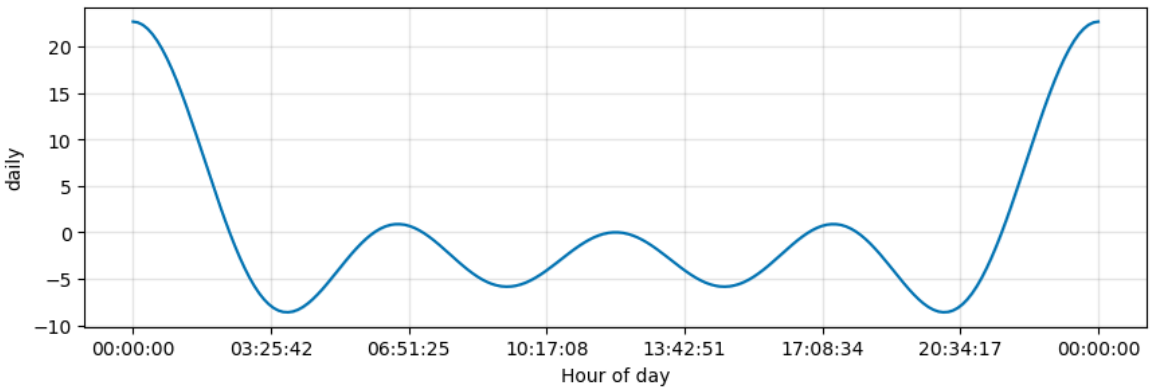


In [22]:


trend
500
0
−500
−1000
−1500
−2000
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
2024-03
ds


weekly
3
2
1
0
−1
−2
−3
−4
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Day of week


yearly
200
100
0
−100
−200
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year


daily
20
15
10
5
0
−5
−10
00:00:00
03:25:42
06:51:25
10:17:08
13:42:51
17:08:34
20:34:17
00:00:00
Hour of day

In [23]:
```
PLOT_COLUMS = [
    "Price",
    "Price (forecast)",
]

mpl.style.use("seaborn")

result_df = prophet_data.copy()
print(result_df.tail(1).rename(columns = {"y": "yhat"}))

# Add first result from forecast as y to connect dots
result_df = result_df.append(result_df.tail(1).rename(columns = {"y": "yhat"}))

result_df = result_df.append(forecast)

result_df = result_df.rename(columns = {
    "ds": "Date",
    "y": "Price",
    "yhat": "Price (forecast)"
})

fig = plt.figure(dpi=100)

fig.set_facecolor("white")

plot = result_df.plot(x="Date", y=PLOT_COLUMS, figsize=(15, 8), ax=fig.gca())

plot_fig = plot.get_figure()
```

```
             ds  yhat
2554 2016-03-10  11.2
```