```
import pandas as pd
import yfinance as yf
import datetime
from datetime import date, timedelta
today = date.today()

d1 = today.strftime("%Y-%m-%d")
end_date = d1
d2 = date.today() - timedelta(days=366)
d2 = d2.strftime("%Y-%m-%d")
start_date = d2

data = yf.download('BTC-USD',
                   start=start_date,
                   end=end_date,
                   progress=False)
data["Date"] = data.index
data = data[["Date", "Open", "High", "Low", "Close", "Adj Close", "Volume"]]
data.reset_index(drop=True, inplace=True)
print(data.head())

print("DATA TAIL PART:")
print(data.tail())

print("DATA DESCRIPTION:")
print(data.describe())

print("DATA INFORMATION:")
print(data.info())



import plotly.graph_objects as go
figure = go.Figure(data=[go.Candlestick(x=data["Date"],
                                        open=data["Open"],
                                        high=data["High"],
                                        low=data["Low"],
                                        close=data["Close"])])
figure.update_layout(title = "Bitcoin Price Analysis",
                     xaxis_rangeslider_visible=False)
figure.show()

correlation = data.corr()
print(correlation["Close"].sort_values(ascending=False))



data.tail()
```

```
print('Total number of days present in the dataset:',data.shape[0])
print('Total number of fields present in the dataset:',data.shape[1])
import pandas as pd
import plotly.graph_objects as go
import yfinance as yf

# Download data for Bitcoin from Yahoo Finance
btc_data = yf.download('BTC-USD', start='2023-05-09', end='2024-05-09')

# Extract open and close prices from the downloaded data
open_prices = btc_data['Open']
close_prices = btc_data['Close']

# Assuming you want to create a bar plot using plotly
fig = go.Figure()
fig.add_trace(go.Bar(
   x=open_prices.index.strftime('%B'),
   y=open_prices,
   name='Bitcoin open price',
   marker_color='crimson'
))
fig.add_trace(go.Bar(
   x=close_prices.index.strftime('%B'),
   y=close_prices,
   name='Bitcoin close price',
   marker_color='lightsalmon'
))
fig.update_layout(
   barmode='group',
   xaxis_tickangle=-45,
   title='Monthwise comparison between stock open and close price'
)

fig.show()

import plotly.graph_objects as go
import yfinance as yf
from itertools import cycle

# Fetch stock data using yfinance
stock_data = yf.download('AAPL', start='2023-05-09', end='2024-05-09')

# Create a sample figure object
fig = go.Figure()

# Add traces to the figure using the fetched stock data
fig.add_trace(go.Scatter(x=stock_data.index, y=stock_data['Open'], mode='lines', name='Open
```

```
Price'))
fig.add_trace(go.Scatter(x=stock_data.index, y=stock_data['Close'], mode='lines', name='Close
Price'))
fig.add_trace(go.Scatter(x=stock_data.index, y=stock_data['High'], mode='lines', name='High
Price'))
fig.add_trace(go.Scatter(x=stock_data.index, y=stock_data['Low'], mode='lines', name='Low
Price'))

# Define the names for the traces using itertools.cycle
names = cycle(['Bitcoin Stock Open Price', 'Bitcoin Stock Close Price', 'Bitcoin Stock High Price',
'Bitcoin Stock Low Price'])

# Update the layout of the figure
fig.update_layout(title_text='Bitcoin Stock Analysis Chart', font_size=15, font_color='black',
legend_title_text='Stock Parameters')

# Assign names to each trace
for trace, name in zip(fig.data, names):
    trace.name = name

# Update axes properties
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

# Show the figure
fig.show()


import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import numpy as np
import yfinance as yf

# Fetch stock data using yfinance
stock_data = yf.download('AAPL', start='2023-05-09', end='2024-05-09')

# Extract 'Close' prices from the fetched data
closedf = stock_data[['Close']].copy()

# Further operations on closedf
scaler = MinMaxScaler(feature_range=(0, 1))
closedf_scaled = scaler.fit_transform(np.array(closedf).reshape(-1, 1))

# You can proceed with your further operations on closedf_scaled if needed
print(closedf_scaled)

import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

```
import numpy as np
import yfinance as yf
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Fetch stock data using yfinance
stock_data = yf.download('AAPL', start='2024-01-01', end='2024-01-03')

# Extract 'Close' prices from the fetched data
closedf = stock_data[['Close']].copy()

# Further operations on closedf
scaler = MinMaxScaler(feature_range=(0, 1))
closedf_scaled = scaler.fit_transform(np.array(closedf).reshape(-1, 1))

# Define the LSTM model
model = Sequential()
model.add(LSTM(10, input_shape=(None, 1), activation="relu"))
model.compile(loss="mean_squared_error", optimizer="adam")
model.add(Dense(1))

# You can proceed with your further operations on the model if needed
print(model.summary())
data = yf.download('AAPL', start='2023-05-09', end='2024-05-09')

# Extract the features and target variable
X_train = data[['Open', 'High', 'Low', 'Close', 'Volume']].values
Y_train = data['Close'].values.reshape(-1, 1)
history = model.fit(X_train, Y_train, epochs=200, batch_size=32, verbose=1)



import yfinance as yf
import matplotlib.pyplot as plt

# Download Bitcoin price data
bitcoin_data = yf.download('BTC-USD', start='2024-04-15', end='2024-05-15')

# Extract last 15 days and next 30 days of Bitcoin prices
last_15_days = bitcoin_data['Close'][-15:]
next_30_days = bitcoin_data['Close'][:30]

# Plot the comparison
plt.figure(figsize=(10, 6))
plt.plot(last_15_days.index, last_15_days.values, label='Last 15 Days', marker='o')
plt.plot(next_30_days.index, next_30_days.values, label='Next 30 Days', marker='o')
plt.xlabel('Date')
plt.ylabel('Bitcoin Price (USD)')
plt.title('Comparison of Last 15 Days vs Next 30 Days Bitcoin Prices')
```

```
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```