AUTOMATED EXTRACTION OF LIVE DATA FROM THE STOCK WEBSITE

AIM: To create a python program for automated extraction of live data from the stock website, visualize the data and perform sentiment analysis.

SOURCE CODE:

```
import requests

from bs4 import BeautifulSoup

import time

import matplotlib.pyplot as plt

def get_sentiment(current_price, previous_price):

    if current_price > previous_price:

        return "Positive"

    elif current_price < previous_price:

        return "Negative"

    else:

        return "Neutral"

def scrape_price(url):

    try:

        response = requests.get(url)

        response.raise_for_status()

        soup = BeautifulSoup(response.text, 'html.parser')

        price_text = soup.find('div', {'class': 'YMlKec fxKbKc'}).text

        price = float(price_text.strip()[1:].replace(",", ""))

        return price

except Exception as e:
```

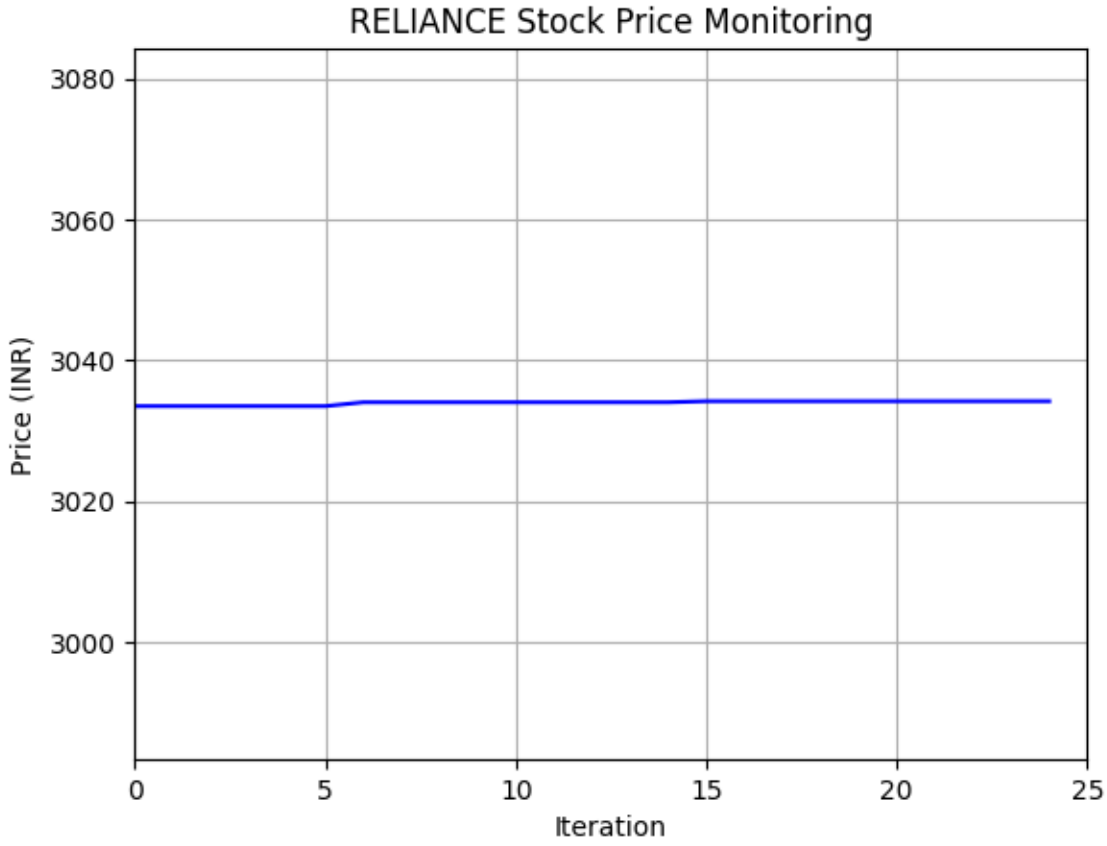
```
        print(f"Error scrapping price: {e}")
        return None
ticker = 'RELIANCE'
url = 'https://www.google.com/finance/quote/RELIANCE:NSE'
n = 100
sleep_time = 0.5
show_sentiment = True
plt.ion()
fig, ax = plt.subplots()
x, y = [], []
line, = ax.plot(x, y, color='b', label='Price Line')  # Line plot
ax.set_xlim(0, n)
previous_price = None
for i in range(n):
    price = scrape_price(url)
    if price is None:
        continue
    if previous_price is not None:
        sentiment = get_sentiment(price, previous_price)
        if show_sentiment:
            print(f"Iteration {i+1}: Price = {price}, Sentiment = {sentiment}")
    else:
        print(f"Iteration {i+1}: Price = {price}, Sentiment = N/A (First Data Point)")
```

```
 previous_price = price
    x.append(i)
    y.append(price)
    line.set_xdata(x)
    line.set_ydata(y)
    #ax.scatter(x, y, color='r')
    ax.set_ylim(min(y) - 50, max(y) + 50)
    fig.canvas.draw()
    fig.canvas.flush_events()
    time.sleep(sleep_time)
plt.xlabel('Iteration')
plt.ylabel('Price (INR)')
plt.title(f'{ticker} Stock Price Monitoring')
plt.grid(True)
plt.ioff()
plt.show()
```

OUTPUT:

```
Iteration 1: Price = 3033.5, Sentiment = N/A (First Data Point)
Iteration 2: Price = 3033.5, Sentiment = Neutral
Iteration 3: Price = 3033.5, Sentiment = Neutral
Iteration 4: Price = 3033.5, Sentiment = Neutral
Iteration 5: Price = 3033.5, Sentiment = Neutral
Iteration 6: Price = 3033.5, Sentiment = Neutral
Iteration 7: Price = 3034.05, Sentiment = Positive
Iteration 8: Price = 3034.05, Sentiment = Neutral
Iteration 9: Price = 3034.05, Sentiment = Neutral
Iteration 10: Price = 3034.05, Sentiment = Neutral
Iteration 11: Price = 3034.05, Sentiment = Neutral
Iteration 12: Price = 3034.05, Sentiment = Neutral
Iteration 13: Price = 3034.05, Sentiment = Neutral
Iteration 14: Price = 3034.05, Sentiment = Neutral
Iteration 15: Price = 3034.05, Sentiment = Neutral
```

```
Iteration 16: Price = 3034.2, Sentiment = Positive
Iteration 17: Price = 3034.2, Sentiment = Neutral
Iteration 18: Price = 3034.2, Sentiment = Neutral
Iteration 19: Price = 3034.2, Sentiment = Neutral
Iteration 20: Price = 3034.2, Sentiment = Neutral
Iteration 21: Price = 3034.2, Sentiment = Neutral
Iteration 22: Price = 3034.2, Sentiment = Neutral
Iteration 23: Price = 3034.2, Sentiment = Neutral
Iteration 24: Price = 3034.2, Sentiment = Neutral
Iteration 25: Price = 3034.2, Sentiment = Neutral
```



EXPLANATION:

1. **Library Imports**:
   - The script uses requests to fetch a webpage containing stock price information.
   - BeautifulSoup (from the bs4 library) parses the HTML content to extract the stock price.
   - time controls the timing between each data collection.
   - matplotlib.pyplot is used to plot the stock prices in real time.
2. **Sentiment Analysis**:
   - The script includes a function to compare the current stock price with the previous one. This function determines whether the stock price has increased, decreased, or remained the same, labeling the sentiment as "Positive," "Negative," or "Neutral."

3. **Price Fetching**:
    - Another function in the script retrieves the stock price from a specified URL. It parses the webpage content to extract the price and converts it into a numerical format. If the process fails, an error message is displayed.
4. **Initialization**:
    - The script sets up variables for the stock ticker symbol, the URL of the Google Finance page, the number of data points to collect, the time interval between each fetch, and a flag to control the display of sentiment analysis.
5. **Plot Setup**:
    - An interactive plot is prepared to display the stock price trend. The plot is initialized with empty data, and its x-axis is configured to handle the specified number of data points.
6. **Main Loop**:
    - The script enters a loop that runs for a specified number of iterations. In each iteration, the stock price is fetched and compared to the previous price to determine the sentiment. The current price and sentiment are printed, and the data is added to the plot. The plot is then updated in real time to reflect the latest price trend.
7. **Final Plot Adjustments**:
    - After all iterations are complete, the plot is finalized with labels, a title, and a grid for better readability. The interactive mode is turned off to keep the plot visible after the loop finishes.

APPLICATIONS:

Real-Time Stock Monitoring:

- **Individual Investors**: Investors can use this script to monitor stock prices in real-time, helping them make informed decisions on buying or selling based on price trends.
- **Day Trading**: Day traders can utilize the script to track intraday price movements and respond quickly to market changes.

Sentiment Analysis:

- **Market Sentiment Tracking**: By integrating real-time sentiment analysis based on price changes, traders can gauge market sentiment. For instance, a series of positive sentiments may indicate a bullish trend, while negative sentiments may signal bearish behavior.
- **Algorithmic Trading**: This script can be extended for algorithmic trading strategies where trades are executed based on real-time sentiment analysis.

Automated Stock Alerts:

- **Price Alerts**: By adding conditional statements, the script could be adapted to send alerts (e.g., via email or SMS) when the stock price reaches a certain threshold.
- **Sentiment-Based Alerts**: Traders can receive notifications when a significant sentiment shift occurs, prompting them to take action.

Customizable Dashboards:

- **Personal Finance Dashboards**: The script can be integrated into a larger dashboard that monitors multiple stocks or other financial metrics, providing a holistic view of one's investment portfolio.
- **Integration with Financial Apps**: The script could be part of a broader application that combines real-time data with other financial tools, like portfolio tracking or risk assessment models.

Research and Analysis:

- **Academic Research**: Researchers studying market dynamics can use this script to observe how stock prices react to various external factors (e.g., news events, earnings reports).
- **Behavioral Finance**: The sentiment analysis aspect can be used to study the psychological factors driving market behavior and how investors react to price changes.

RESULT: A python program is implemented to extract live data from stock market website, visualize the data and sentiment analysis is also performed.