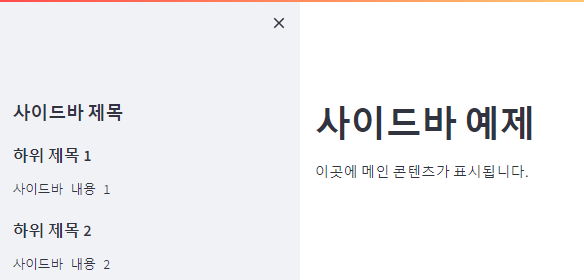
사이드 바 예제 코드

|  |
| --- |
| 변경 전 |
| # 필요한 모듈 가져오기  import streamlit as st  # Streamlet 애플리케이션 시작  st.title("사이드바 예제")  # 사이드바 생성  st.sidebar.header("사이드바 제목")  st.sidebar.subheader("하위 제목 1")  st.sidebar.text("사이드바 내용 1")  st.sidebar.subheader("하위 제목 2")  st.sidebar.text("사이드바 내용 2")  # 메인 콘텐츠 영역  st.write("이곳에 메인 콘텐츠가 표시됩니다.") |
| 변경 후 |
| # 필요한 모듈 가져오기  import streamlit as st  # Streamlet 애플리케이션 시작  st.title("사이드바 예제")  # 사이드바 생성  st.sidebar.header("사이드바 제목")  st.sidebar.subheader("하위 제목 1")  st.sidebar.text("사이드바 내용 1")  st.sidebar.subheader("하위 제목 2")  st.sidebar.text("사이드바 내용 2")  # 메인 콘텐츠 영역  st.write("이곳에 메인 콘텐츠가 표시됩니다.") |



레이아웃 예제 코드

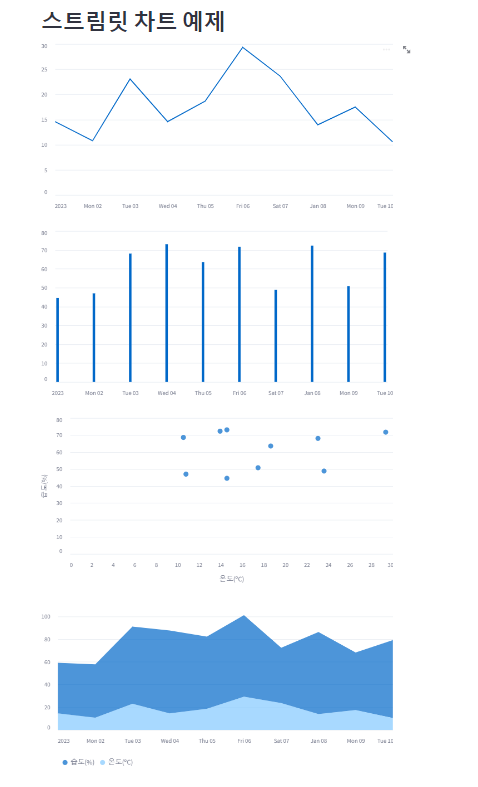
columns 과 with 사용

|  |
| --- |
| # 필요한 모듈 가져오기  import streamlit as st  # 레이아웃 설정  st.title("스트림릿 레이아웃 예제")  # 첫 번째 행 (Row)  col1, col2, col3 = st.columns((1, 1, 1))  # 첫 번째 열 (Column)  with col1:  st.header("첫 번째 열")  st.image("https://images.unsplash.com/photo-1548407260-da850faa41e3?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1487&q=80")  st.write("이 열에 대한 내용을 작성합니다.")  # 두 번째 열 (Column)  with col2:  st.header("두 번째 열")  st.image("https://images.unsplash.com/photo-1548407260-da850faa41e3?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1487&q=80")  st.write("이 열에 대한 내용을 작성합니다.")  # 세 번째 열 (Column)  with col3:  st.header("세 번째 열")  st.image("https://images.unsplash.com/photo-1548407260-da850faa41e3?ixlib=rb-1.2.1&ixid=eyJhcHBfaWQiOjEyMDd9&auto=format&fit=crop&w=1487&q=80")  st.write("이 열에 대한 내용을 작성합니다.")  # 두 번째 행 (Row)  st.write("두 번째 행에 대한 내용을 작성합니다.")  # 세 번째 행 (Row)  st.subheader("세 번째 행")  st.write("세 번째 행에 대한 내용을 작성합니다.") |



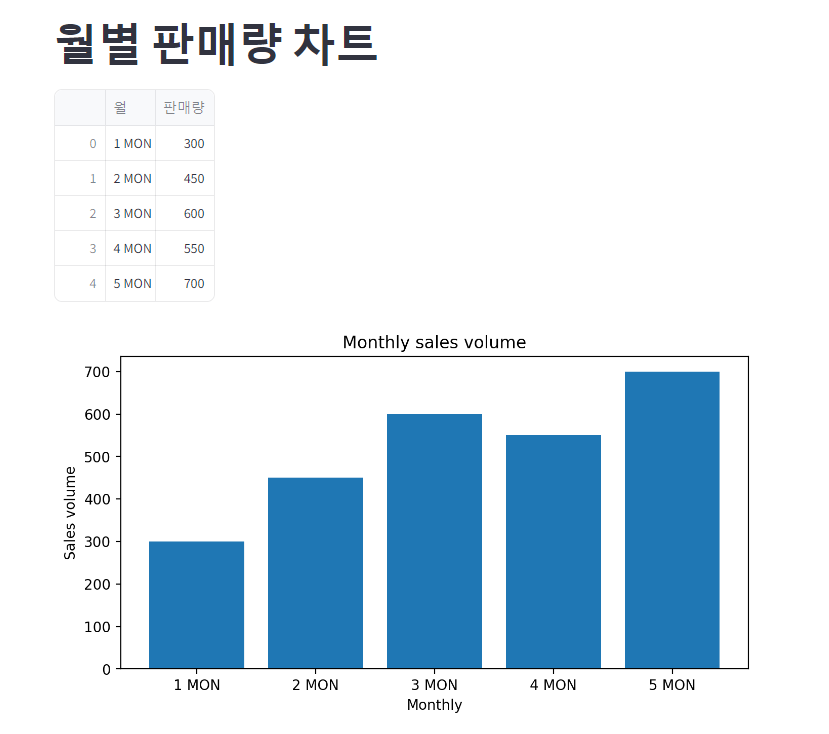
차트 그리기

|  |
| --- |
| # 필요한 모듈 가져오기  import streamlit as st  import pandas as pd  import numpy as np  # 데이터 생성  data = {  '날짜': pd.date\_range(start="2023-01-01", periods=10, freq='D'),  '온도(ºC)': np.random.uniform(10, 30, 10),  '습도(%)': np.random.uniform(40, 80, 10)  }  df = pd.DataFrame(data)  # 레이아웃 설정  st.title("스트림릿 차트 예제")  # 선 그래프  st.line\_chart(df.set\_index('날짜')['온도(ºC)'], use\_container\_width=True)  # 막대 그래프  st.bar\_chart(df.set\_index('날짜')['습도(%)'], use\_container\_width=True)  # 산점도  st.scatter\_chart(df, x='온도(ºC)', y='습도(%)')  # 아름다운 차트  st.area\_chart(df.set\_index('날짜')[['온도(ºC)', '습도(%)']], use\_container\_width=True) |



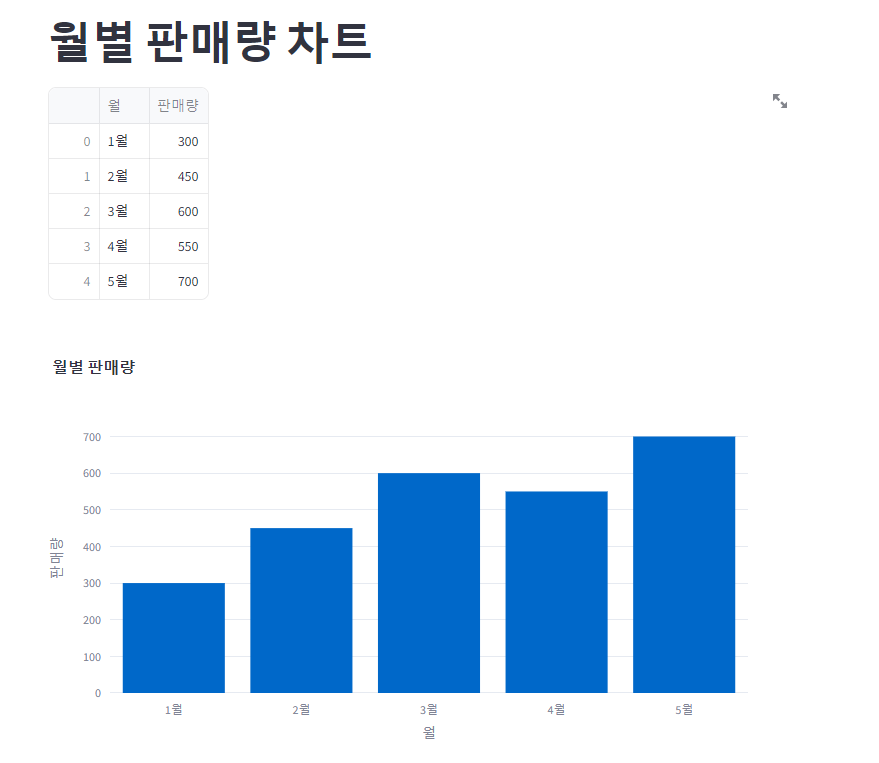
matplotlib를 이용해서 차트 예제 코드

|  |
| --- |
| import streamlit as st  import pandas as pd  import matplotlib.pyplot as plt  import numpy as np  # 데이터 생성  data = {  '월': ['1 MON', '2 MON', '3 MON', '4 MON', '5 MON'],  '판매량': [300, 450, 600, 550, 700]  }  df = pd.DataFrame(data)  # 스트림릿 애플리케이션 시작  st.title("월별 판매량 차트")  # 데이터프레임 출력  st.write(df)  # Matplotlib을 사용하여 차트 생성  plt.figure(figsize=(8, 4))  plt.bar(df['월'], df['판매량'])  plt.xlabel('Monthly')  plt.ylabel('Sales volume')  plt.title('Monthly sales volume')  # 생성한 차트를 스트림릿 애플리케이션에 추가  st.pyplot(plt) |



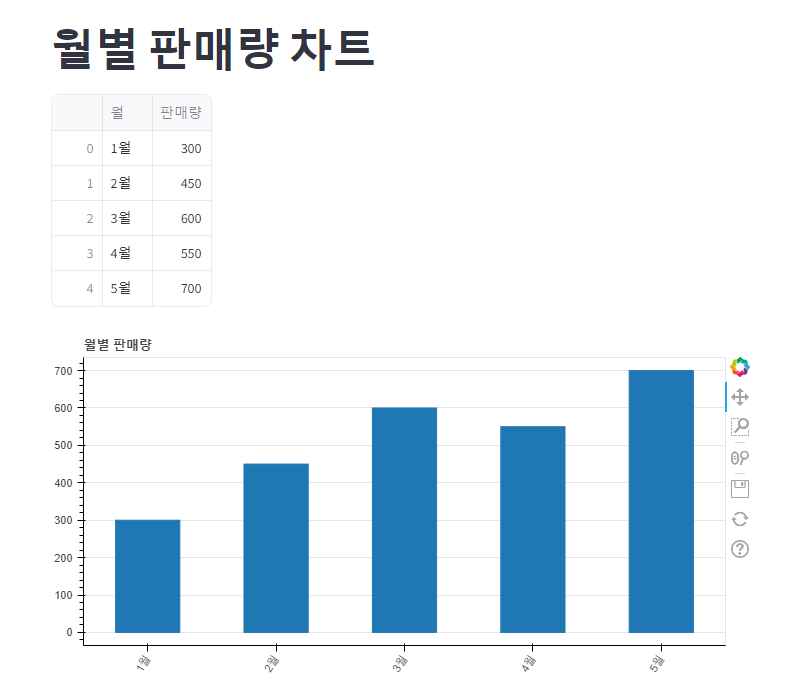
Plotly(인터랙티브 차트)

|  |
| --- |
| import streamlit as st  import pandas as pd  import plotly.express as px  # 데이터 생성  data = {  '월': ['1월', '2월', '3월', '4월', '5월'],  '판매량': [300, 450, 600, 550, 700]  }  df = pd.DataFrame(data)  # 스트림릿 애플리케이션 시작  st.title("월별 판매량 차트")  # 데이터프레임 출력  st.write(df)  # Plotly를 사용하여 막대 그래프 생성  fig = px.bar(df, x='월', y='판매량', title='월별 판매량')  st.plotly\_chart(fig) |



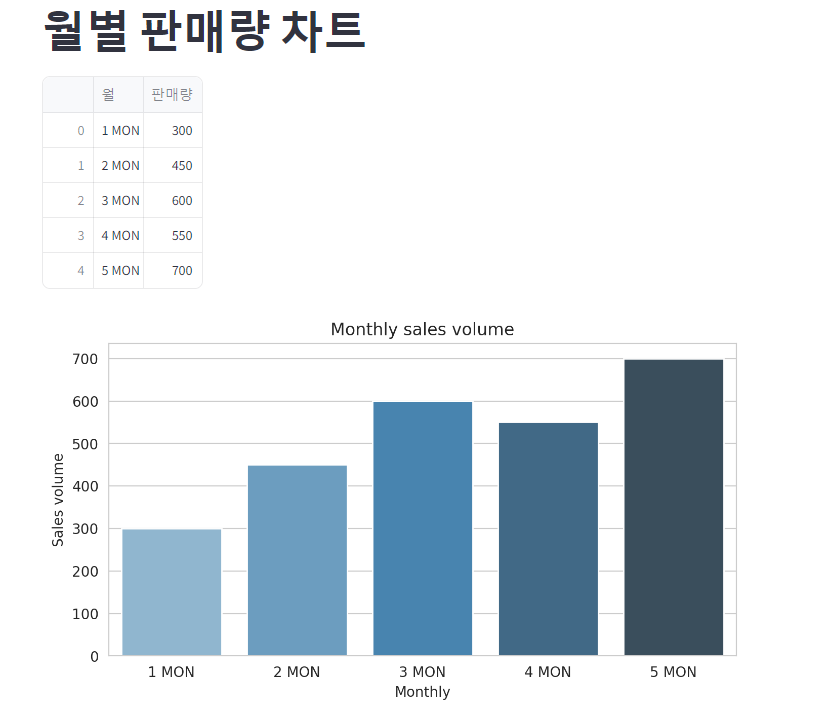
Bokeh 차트 그리는 코드

|  |
| --- |
| import streamlit as st  import pandas as pd  from bokeh.plotting import figure  from bokeh.models import ColumnDataSource  # 데이터 생성  data = {  '월': ['1월', '2월', '3월', '4월', '5월'],  '판매량': [300, 450, 600, 550, 700]  }  df = pd.DataFrame(data)  # 스트림릿 애플리케이션 시작  st.title("월별 판매량 차트")  # 데이터프레임 출력  st.write(df)  # Bokeh를 사용하여 막대 그래프 생성  source = ColumnDataSource(data=df)  p = figure(x\_range=df['월'], plot\_height=350, title="월별 판매량")  p.vbar(x='월', top='판매량', width=0.5, source=source)  p.xaxis.major\_label\_orientation = 1  p.xgrid.grid\_line\_color = None  st.bokeh\_chart(p, use\_container\_width=True) |



Seaborn 차트 예제 코드

|  |
| --- |
| import streamlit as st  import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  # 데이터 생성  data = {  '월': ['1 MON', '2 MON', '3 MON', '4 MON', '5 MON'],  '판매량': [300, 450, 600, 550, 700]  }  df = pd.DataFrame(data)  # 스트림릿 애플리케이션 시작  st.title("월별 판매량 차트")  # 데이터프레임 출력  st.write(df)  # Seaborn을 사용하여 막대 그래프 생성  sns.set\_style("whitegrid")  plt.figure(figsize=(8, 4))  sns.barplot(data=df, x='월', y='판매량', palette="Blues\_d")  plt.xlabel('Monthly')  plt.ylabel('Sales volume')  plt.title('Monthly sales volume')  st.pyplot(plt) |



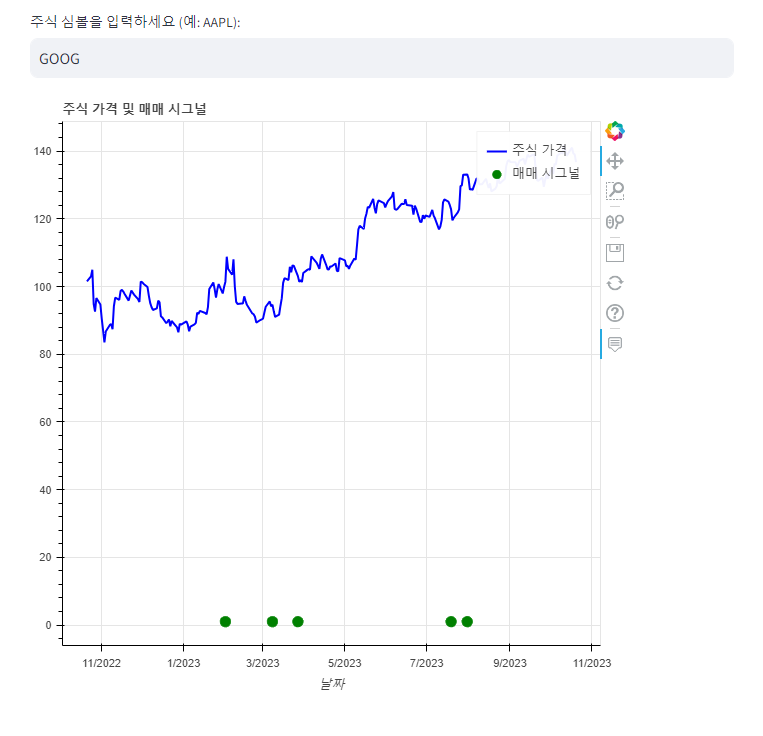
|  |
| --- |
| # 필요한 라이브러리를 불러옵니다.  import streamlit as st  import yfinance as yf  from bokeh.plotting import figure, show  from bokeh.models import ColumnDataSource  from bokeh.io import curdoc  from bokeh.layouts import layout  from bokeh.models.tools import HoverTool  import pandas as pd  from datetime import datetime  # 스트림릿 앱 제목을 설정합니다.  st.title('주식 가격 및 이동평균')  # 주식 심볼을 입력받습니다.  stock\_symbol = st.text\_input('주식 심볼을 입력하세요 (예: AAPL):')  if not stock\_symbol:  st.stop()  # Yahoo Finance에서 주식 데이터를 불러옵니다.  stock\_data = yf.Ticker(stock\_symbol)  # 주식 데이터를 가져옵니다.  stock\_df = stock\_data.history(period="1y")  # 50일 이동평균을 계산합니다.  stock\_df['50일 이동평균'] = stock\_df['Close'].rolling(window=50).mean()  # Bokeh 도구 상자 설정  hover = HoverTool(  tooltips=[  ("날짜", "@Date{%F}"),  ("가격", "@Close{$0.2f}"),  ("50일 이동평균", "@{50일 이동평균}{$0.2f}")  ],  formatters={"Date": "datetime"},  mode='vline'  )  # Bokeh 도화지 생성  p = figure(title='주식 가격 및 50일 이동평균', x\_axis\_label='날짜', x\_axis\_type='datetime')  # 주식 가격과 50일 이동평균 데이터를 소스로 설정  source = ColumnDataSource(stock\_df)  # 주식 가격 그래프 추가  p.line(x='Date', y='Close', source=source, legend\_label='주식 가격', line\_color='blue', line\_width=2)  # 50일 이동평균 그래프 추가  p.line(x='Date', y='50일 이동평균', source=source, legend\_label='50일 이동평균', line\_color='orange', line\_width=2)  # 그래프에 도구 상자 추가  p.add\_tools(hover)  # 그래프를 표시  st.bokeh\_chart(p, use\_container\_width=True)  # 이동평균 크로스 검사  for i in range(1, len(stock\_df)):  if stock\_df['50일 이동평균'][i] > stock\_df['Close'][i] and stock\_df['50일 이동평균'][i-1] <= stock\_df['Close'][i-1]:  st.write(f"50일 이동평균이 주식 가격 아래에서 위로 크로스되었습니다. 이것은 지지 수준으로 간주할 수 있습니다. 날짜: {stock\_df.index[i]}")  elif stock\_df['50일 이동평균'][i] < stock\_df['Close'][i] and stock\_df['50일 이동평균'][i-1] >= stock\_df['Close'][i-1]:  st.write(f"50일 이동평균이 주식 가격 위에서 아래로 크로스되었습니다. 이것은 저항 수준으로 간주할 수 있습니다. 날짜: {stock\_df.index[i]}")  # 주식 데이터를 표로 출력합니다.  st.write(stock\_df) |



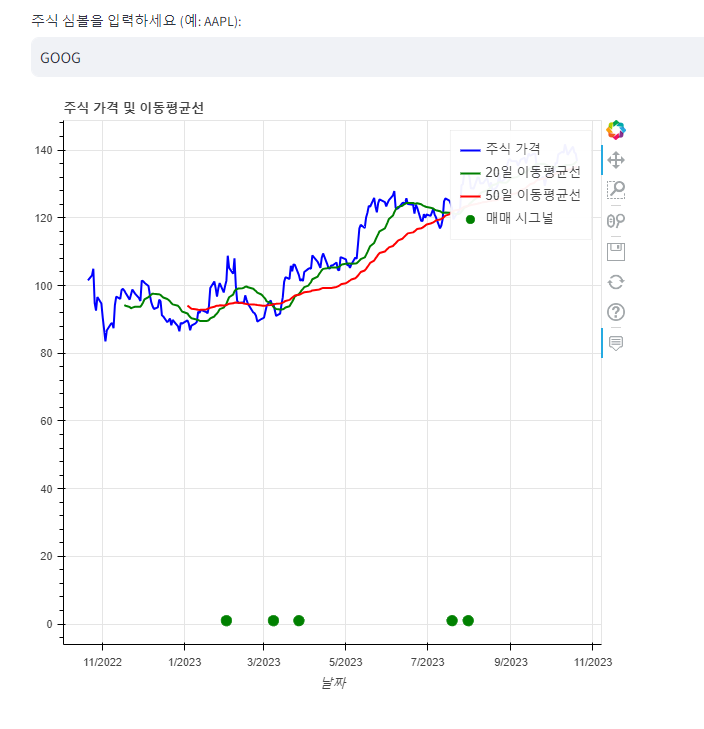
|  |
| --- |
| import yfinance as yf  import pandas as pd  import streamlit as st  # 주식 심볼을 입력받습니다.  stock\_symbol = st.text\_input('주식 심볼을 입력하세요 (예: AAPL):')  # Yahoo Finance에서 주식 데이터를 불러옵니다.  stock\_data = yf.Ticker(stock\_symbol)  # 주식 데이터를 가져옵니다.  stock\_df = stock\_data.history(period="1y")  # 단기 이동평균선(20일)과 장기 이동평균선(50일)을 계산합니다.  stock\_df['20일 이동평균'] = stock\_df['Close'].rolling(window=20).mean()  stock\_df['50일 이동평균'] = stock\_df['Close'].rolling(window=50).mean()  # 매매 시그널 초기화  매매\_시그널 = 0  매매\_시그널\_날짜 = [] # 매매 시그널 발생 날짜를 저장할 리스트  # 매매 시그널 생성  for i in range(1, len(stock\_df)):  if stock\_df['20일 이동평균'][i] > stock\_df['50일 이동평균'][i] and stock\_df['20일 이동평균'][i-1] <= stock\_df['50일 이동평균'][i-1]:  매매\_시그널 = 1 # 매수 신호 생성  매매\_시그널\_날짜.append(stock\_df.index[i]) # 매수 신호 발생 날짜 저장  elif stock\_df['20일 이동평균'][i] < stock\_df['50일 이동평균'][i] and stock\_df['20일 이동평균'][i-1] >= stock\_df['50일 이동평균'][i-1]:  매매\_시그널 = -1 # 매도 신호 생성  매매\_시그널\_날짜.append(stock\_df.index[i]) # 매도 신호 발생 날짜 저장  else:  매매\_시그널 = 0 # 신호 없음  # 매매 시그널을 데이터프레임에 추가  stock\_df.at[stock\_df.index[i], '매매\_시그널'] = 매매\_시그널  # 매매 시그널이 1이면 매수, -1이면 매도, 0이면 홀딩  st.write(stock\_df[['Close', '20일 이동평균', '50일 이동평균', '매매\_시그널']])  # 매매 시그널이 발생한 날짜 출력  st.write("매매 시그널이 발생한 날짜:", 매매\_시그널\_날짜) |



|  |
| --- |
| import yfinance as yf  import pandas as pd  from bokeh.plotting import figure, show  from bokeh.models import ColumnDataSource  from bokeh.models.tools import HoverTool  from datetime import datetime  import streamlit as st  # 주식 심볼을 입력받습니다.  stock\_symbol = st.text\_input('주식 심볼을 입력하세요 (예: AAPL):')  # Yahoo Finance에서 주식 데이터를 불러옵니다.  stock\_data = yf.Ticker(stock\_symbol)  # 주식 데이터를 가져옵니다.  stock\_df = stock\_data.history(period="1y")  # 단기 이동평균선(20일)과 장기 이동평균선(50일)을 계산합니다.  stock\_df['20일 이동평균'] = stock\_df['Close'].rolling(window=20).mean()  stock\_df['50일 이동평균'] = stock\_df['Close'].rolling(window=50).mean()  # 매매 시그널 초기화  매매\_시그널 = 0  매매\_시그널\_날짜 = [] # 매매 시그널 발생 날짜를 저장할 리스트  # 매매 시그널 생성  for i in range(1, len(stock\_df)):  if stock\_df['20일 이동평균'][i] > stock\_df['50일 이동평균'][i] and stock\_df['20일 이동평균'][i-1] <= stock\_df['50일 이동평균'][i-1]:  매매\_시그널 = 1 # 매수 신호 생성  매매\_시그널\_날짜.append(stock\_df.index[i]) # 매수 신호 발생 날짜 저장  elif stock\_df['20일 이동평균'][i] < stock\_df['50일 이동평균'][i] and stock\_df['20일 이동평균'][i-1] >= stock\_df['50일 이동평균'][i-1]:  매매\_시그널 = -1 # 매도 신호 생성  매매\_시그널\_날짜.append(stock\_df.index[i]) # 매도 신호 발생 날짜 저장  else:  매매\_시그널 = 0 # 신호 없음  # 매매 시그널을 데이터프레임에 추가  stock\_df.at[stock\_df.index[i], '매매\_시그널'] = 매매\_시그널  # Bokeh 그래프 설정  p = figure(title='주식 가격 및 매매 시그널', x\_axis\_label='날짜', x\_axis\_type='datetime')  # 주식 가격 그래프 추가  p.line(x=stock\_df.index, y=stock\_df['Close'], legend\_label='주식 가격', line\_color='blue', line\_width=2)  # 매매 시그널을 표시하는 그래프 추가  매매\_시그널\_소스 = ColumnDataSource(data={'날짜': 매매\_시그널\_날짜, '매매\_시그널': [1]\*len(매매\_시그널\_날짜)})  p.circle(x='날짜', y='매매\_시그널', source=매매\_시그널\_소스, size=10, color='green', legend\_label='매매 시그널')  # 도구 상자 추가  hover = HoverTool(tooltips=[("날짜", "@날짜{%F}"), ("가격", "@y{$0.2f}")], formatters={"날짜": "datetime"}, mode='vline')  p.add\_tools(hover)  # 그래프 출력  st.bokeh\_chart(p) |



|  |
| --- |
| import yfinance as yf  import pandas as pd  from bokeh.plotting import figure, show  from bokeh.models import ColumnDataSource  from bokeh.models.tools import HoverTool  from datetime import datetime  import streamlit as st  # 주식 심볼을 입력받습니다.  stock\_symbol = st.text\_input('주식 심볼을 입력하세요 (예: AAPL):')  # Yahoo Finance에서 주식 데이터를 불러옵니다.  stock\_data = yf.Ticker(stock\_symbol)  # 주식 데이터를 가져옵니다.  stock\_df = stock\_data.history(period="1y")  # 단기 이동평균선(20일)과 장기 이동평균선(50일)을 계산합니다.  stock\_df['20일 이동평균'] = stock\_df['Close'].rolling(window=20).mean()  stock\_df['50일 이동평균'] = stock\_df['Close'].rolling(window=50).mean()  # 매매 시그널 초기화  매매\_시그널 = 0  매매\_시그널\_날짜 = [] # 매매 시그널 발생 날짜를 저장할 리스트  # 매매 시그널 생성  for i in range(1, len(stock\_df)):  if stock\_df['20일 이동평균'][i] > stock\_df['50일 이동평균'][i] and stock\_df['20일 이동평균'][i-1] <= stock\_df['50일 이동평균'][i-1]:  매매\_시그널 = 1 # 매수 신호 생성  매매\_시그널\_날짜.append(stock\_df.index[i]) # 매수 신호 발생 날짜 저장  elif stock\_df['20일 이동평균'][i] < stock\_df['50일 이동평균'][i] and stock\_df['20일 이동평균'][i-1] >= stock\_df['50일 이동평균'][i-1]:  매매\_시그널 = -1 # 매도 신호 생성  매매\_시그널\_날짜.append(stock\_df.index[i]) # 매도 신호 발생 날짜 저장  else:  매매\_시그널 = 0 # 신호 없음  # 매매 시그널을 데이터프레임에 추가  stock\_df.at[stock\_df.index[i], '매매\_시그널'] = 매매\_시그널  # Bokeh 그래프 설정  p = figure(title='주식 가격 및 이동평균선', x\_axis\_label='날짜', x\_axis\_type='datetime')  # 주식 가격 그래프 추가  p.line(x=stock\_df.index, y=stock\_df['Close'], legend\_label='주식 가격', line\_color='blue', line\_width=2)  # 20일 이동평균선 그래프 추가  p.line(x=stock\_df.index, y=stock\_df['20일 이동평균'], legend\_label='20일 이동평균선', line\_color='green', line\_width=2)  # 50일 이동평균선 그래프 추가  p.line(x=stock\_df.index, y=stock\_df['50일 이동평균'], legend\_label='50일 이동평균선', line\_color='red', line\_width=2)  # 매매 시그널을 표시하는 그래프 추가  매매\_시그널\_소스 = ColumnDataSource(data={'날짜': 매매\_시그널\_날짜, '매매\_시그널': [1]\*len(매매\_시그널\_날짜)})  p.circle(x='날짜', y='매매\_시그널', source=매매\_시그널\_소스, size=10, color='green', legend\_label='매매 시그널')  # 도구 상자 추가  hover = HoverTool(tooltips=[("날짜", "@날짜{%F}"), ("가격", "@y{$0.2f}")], formatters={"날짜": "datetime"}, mode='vline')  p.add\_tools(hover)  # 그래프 출력  st.bokeh\_chart(p) |



|  |
| --- |
| import yfinance as yf  import pandas as pd  from bokeh.plotting import figure, show  from bokeh.models import ColumnDataSource  from bokeh.models.tools import HoverTool  from datetime import datetime  import streamlit as st  # 주식 심볼을 입력받습니다.  stock\_symbol = st.text\_input('주식 심볼을 입력하세요 (예: AAPL):')  # Yahoo Finance에서 주식 데이터를 불러옵니다.  stock\_data = yf.Ticker(stock\_symbol)  # 주식 데이터를 가져옵니다.  stock\_df = stock\_data.history(period="1y")  # 단기 이동평균선(20일)과 장기 이동평균선(50일)을 계산합니다.  stock\_df['20일 이동평균'] = stock\_df['Close'].rolling(window=20).mean()  stock\_df['50일 이동평균'] = stock\_df['Close'].rolling(window=50).mean()  # 매매 시그널 초기화  매매\_시그널 = 0  매매\_시그널\_날짜 = [] # 매매 시그널 발생 날짜를 저장할 리스트  # 매매 시그널 생성  for i in range(1, len(stock\_df)):  if stock\_df['20일 이동평균'][i] > stock\_df['50일 이동평균'][i] and stock\_df['20일 이동평균'][i-1] <= stock\_df['50일 이동평균'][i-1]:  매매\_시그널 = 1 # 매수 신호 생성  매매\_시그널\_날짜.append(stock\_df.index[i]) # 매수 신호 발생 날짜 저장  elif stock\_df['20일 이동평균'][i] < stock\_df['50일 이동평균'][i] and stock\_df['20일 이동평균'][i-1] >= stock\_df['50일 이동평균'][i-1]:  매매\_시그널 = -1 # 매도 신호 생성  매매\_시그널\_날짜.append(stock\_df.index[i]) # 매도 신호 발생 날짜 저장  else:  매매\_시그널 = 0 # 신호 없음  # 매매 시그널을 데이터프레임에 추가  stock\_df.at[stock\_df.index[i], '매매\_시그널'] = 매매\_시그널  # Bokeh 그래프 설정  p = figure(title='주식 가격 및 이동평균선', x\_axis\_label='날짜', x\_axis\_type='datetime')  # 주식 가격 그래프 추가  p.line(x=stock\_df.index, y=stock\_df['Close'], legend\_label='주식 가격', line\_color='blue', line\_width=2)  # 20일 이동평균선 그래프 추가  p.line(x=stock\_df.index, y=stock\_df['20일 이동평균'], legend\_label='20일 이동평균선', line\_color='green', line\_width=2)  # 50일 이동평균선 그래프 추가  p.line(x=stock\_df.index, y=stock\_df['50일 이동평균'], legend\_label='50일 이동평균선', line\_color='red', line\_width=2)  # 매매 시그널을 표시하는 그래프 추가  매매\_매수 = stock\_df[stock\_df['매매\_시그널'] == 1]  매매\_매도 = stock\_df[stock\_df['매매\_시그널'] == -1]  p.circle(x=매매\_매수.index, y=매매\_매수['Close'], legend\_label='매수 신호', size=10, color='green', fill\_alpha=0.5)  p.circle(x=매매\_매도.index, y=매매\_매도['Close'], legend\_label='매도 신호', size=10, color='red', fill\_alpha=0.5)  # 도구 상자 추가  hover = HoverTool(tooltips=[("날짜", "@x{%F}"), ("가격", "@y{$0.2f}")], formatters={"x": "datetime"}, mode='vline')  p.add\_tools(hover)  # 그래프 출력  st.bokeh\_chart(p) |



|  |
| --- |
| import streamlit as st  import pandas as pd  import matplotlib.pyplot as plt  # Start the Streamlit app  st.title('Stock Price Chart with Support and Resistance Lines')  # Upload a CSV file  uploaded\_file = st.file\_uploader('Upload a CSV file:', type=['csv'])  if uploaded\_file is not None:  # Read the uploaded file into a Pandas DataFrame  data = pd.read\_csv(uploaded\_file)    # Draw the stock price chart  st.write('Stock Price Chart:')  fig, ax = plt.subplots()  ax.plot(data['Date'], data['Close'], label='Price', color='blue')  ax.set\_xlabel('Date')  ax.set\_ylabel('Close')    # Calculate the low and high points  low\_point = data['Close'].min()  high\_point = data['Close'].max()    # Draw support lines  st.write('Support Lines:')  st.write(f'Low Point: {low\_point}')  ax.hlines(low\_point, xmin=data['Date'].min(), xmax=data['Date'].max(), label='Support Line', linestyle='--', color='green')    # Draw resistance lines  st.write('Resistance Lines:')  st.write(f'High Point: {high\_point}')  ax.hlines(high\_point, xmin=data['Date'].min(), xmax=data['Date'].max(), label='Resistance Line', linestyle='--', color='red')    ax.legend()    # Display the chart in Streamlit  st.pyplot(fig) |



|  |
| --- |
| import streamlit as st  import pandas as pd  # 함수 정의 부분 (주어진 함수를 그대로 사용)  #method 1: fractal candlestick pattern  def detect\_level\_method\_1(df):  levels = []  for i in range(2,df.shape[0]-2):  if is\_support(df,i):  l = df['Low'][i]  if is\_far\_from\_level(l, levels, df):  levels.append((i,l))  elif is\_resistance(df,i):  l = df['High'][i]  if is\_far\_from\_level(l, levels, df):  levels.append((i,l))  return levels  #method 2: window shifting method  def detect\_level\_method\_2(df):  levels = []  max\_list = []  min\_list = []  for i in range(5, len(df)-5):  high\_range = df['High'][i-5:i+4]  current\_max = high\_range.max()  if current\_max not in max\_list:  max\_list = []  max\_list.append(current\_max)  if len(max\_list) == 5 and is\_far\_from\_level(current\_max, levels, df):  levels.append((high\_range.idxmax(), current\_max))    low\_range = df['Low'][i-5:i+5]  current\_min = low\_range.min()  if current\_min not in min\_list:  min\_list = []  min\_list.append(current\_min)  if len(min\_list) == 5 and is\_far\_from\_level(current\_min, levels, df):  levels.append((low\_range.idxmin(), current\_min))  return levels  # to detect breakout  def has\_breakout(levels, previous, last):  for \_, level in levels:  cond1 = (previous['Open'] < level)  cond2 = (last['Open'] > level) and (last['Low'] > level)  return (cond1 and cond2)  # 스트림릿 앱 시작  st.title('S&P 500 주식 스크리너')  # S&P 500 기업 목록 가져오기  payload = pd.read\_html('https://en.wikipedia.org/wiki/List\_of\_S%26P\_500\_companies')  stock\_list = payload[0]['Symbol'].values.tolist()  # 기업 목록 선택  selected\_stocks = st.multiselect('S&P 500 기업 선택', stock\_list)  # 선택한 기업 목록을 기반으로 스크리닝  screened\_list\_1 = []  screened\_list\_2 = []  for symbol in selected\_stocks:  try:  df = get\_stock\_price(symbol)    # get levels using the first method  levels\_1 = detect\_level\_method\_1(df)  if (has\_breakout(levels\_1[-5:], df.iloc[-2], df.iloc[-1])):  screened\_list\_1.append(symbol)  # get levels using the second method  levels\_2 = detect\_level\_method\_2(df)  if (has\_breakout(levels\_2[-5:], df.iloc[-2], df.iloc[-1])):  screened\_list\_2.append(symbol)  except Exception as e:  print(e)  # 결과 표시  st.write('Method 1 스크리닝 결과:', screened\_list\_1)  st.write('Method 2 스크리닝 결과:', screened\_list\_2) |

