# Algorithmic Trading

**Algorithmic Trading, also known as Algo Trading or Automated Trading, is the use of advanced computer algorithms to execute trades in financial markets. It has revolutionized the way trading is done by replacing human decision-making with technology-driven processes.**

**By leveraging sophisticated algorithms and mathematical models, algorithmic trading aims to identify trading opportunities, minimize risk, and maximize profits. With the advent of advanced computing power and data analysis tools, algorithmic trading has become increasingly popular in recent years. It has revolutionized the financial industry, providing traders with faster and more accurate ways to execute trades, while also improving market liquidity and efficiency.**

## Algorithmic Trading using Python

**In this project, we will implement two algorithmic trading strategies, namely the momentum strategy and the moving average crossover strategy, on stock price data using Python. Specifically, we will use the stock prices of OPAP, a leading Greek gambling and lottery company, as a reference for our analysis.**

**The momentum strategy involves buying stocks when the momentum is positive and selling them when the momentum is negative, with the goal of achieving superior returns. Meanwhile, the moving average crossover strategy involves analyzing the intersection of short-term and long-term moving averages to predict potential buy and sell signals.**

**By implementing these strategies using Python, we can leverage the power of advanced data analysis tools to improve the accuracy of our predictions and optimize our trading decisions.**

## So let’s import the necessary Python libraries and collect the stock data of OPAP using the yfinance API:

In [1]:
```
import pandas as pd
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import plotly.express as px
import yfinance as yf
import numpy as np
pd.options.mode.chained_assignment = None
```

In [2]:
```
# Get OPAP's stock data from yahoo finance

stock = yf.Ticker("OPAP.AT")
data = stock.history(period="1y")
print(data.head())
```

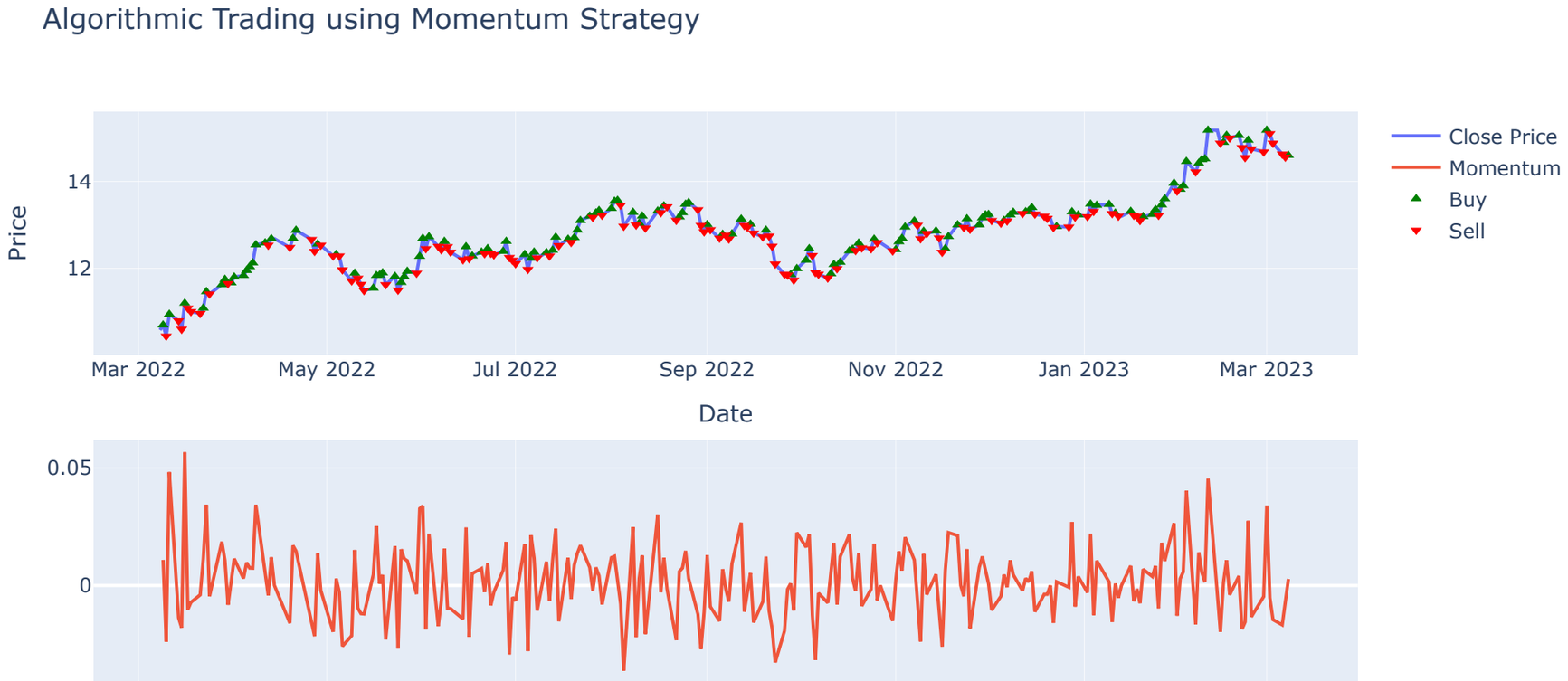
```
                                 Open       High        Low      Close  Volume  \
Date
2022-03-08 00:00:00+02:00  10.737310  10.896382  10.569402  10.578239  817221
2022-03-09 00:00:00+02:00  10.852195  10.940569  10.693125  10.693125  590641
2022-03-10 00:00:00+02:00  10.772659  10.772659  10.357307  10.436843  472823
2022-03-11 00:00:00+02:00  10.551728  11.002430  10.498704  10.940569  435630
2022-03-14 00:00:00+02:00  11.020105  11.073129  10.790336  10.790336  249855

                           Dividends  Stock Splits
Date
2022-03-08 00:00:00+02:00        0.0           0.0
2022-03-09 00:00:00+02:00        0.0           0.0
2022-03-10 00:00:00+02:00        0.0           0.0
2022-03-11 00:00:00+02:00        0.0           0.0
2022-03-14 00:00:00+02:00        0.0           0.0
```

## Now let’s implement the momentum strategy in Algorithmic Trading using Python:

In [3]:
```
# Calculation of momentum

data['momentum'] = data['Close'].pct_change()
```

In [ ]:
```
# Creating subplots to show momentum and buying/selling markers

figure = make_subplots(rows=2, cols=1)
figure.add_trace(go.Scatter(x=data.index,
                            y=data['Close'],
                            name='Close Price'))
figure.add_trace(go.Scatter(x=data.index,
                            y=data['momentum'],
                            name='Momentum',
                            yaxis='y2'))
```

In [5]:
```
# Adding the buy and sell signals

figure.add_trace(go.Scatter(x=data.loc[data['momentum'] > 0].index,
                            y=data.loc[data['momentum'] > 0]['Close'],
                            mode='markers', name='Buy',
                            marker=dict(color='green', symbol='triangle-up')))

figure.add_trace(go.Scatter(x=data.loc[data['momentum'] < 0].index,
                            y=data.loc[data['momentum'] < 0]['Close'],
                            mode='markers', name='Sell',
                            marker=dict(color='red', symbol='triangle-down')))

figure.update_layout(title='Algorithmic Trading using Momentum Strategy',
                     xaxis_title='Date',
                     yaxis_title='Price')
figure.update_yaxes(title="Momentum", secondary_y=True)
figure.show()
```


Algorithmic Trading using Momentum Strategy
Price
14
12
Mar 2022
May 2022
Jul 2022
Sep 2022
Nov 2022
Jan 2023
Mar 2023
Date
Close Price
Momentum
Buy
Sell
0.05
0

**So this is how we can implement an Algorithmic Trading strategy using the momentum strategy. In the above graph, the buy and sell signals are indicated by green triangle-up and red triangle-down markers respectively.**

**For the next step lets try and implement the Moving Average Crossover Strategy using Python:**

In [6]:
```
# Calculating short-term and long-term moving averages

short_window = 50
long_window = 200
data['short_ma'] = data['Close'].rolling(window=short_window, min_periods=1).mean()
data['long_ma'] = data['Close'].rolling(window=long_window, min_periods=1).mean()
```

In [7]:
```
# Creating buy/sell signals

data['signal'] = 0.0
data['signal'][short_window:] = np.where(data['short_ma'][short_window:] > data['long_ma'][short_window:], 1.0, 0.0)

data['positions'] = data['signal'].diff()
```

In [8]:
```
# Creating the plot

figure = make_subplots(rows=2, cols=1, shared_xaxes=True,
                       vertical_spacing=0.02, subplot_titles=("Stock Price", "Moving Averages"))

figure.add_trace(go.Candlestick(x=data.index,
                                open=data['Open'],
                                high=data['High'],
                                low=data['Low'],
                                close=data['Close'], name='Candlestick'), row=1, col=1)

figure.add_trace(go.Scatter(x=data.index, y=data['short_ma'],
                            mode='lines', name='Short-Term MA'), row=2, col=1)

figure.add_trace(go.Scatter(x=data.index, y=data['long_ma'],
                            mode='lines', name='Long-Term MA'), row=2, col=1)

figure.add_trace(go.Scatter(x=data.loc[data['positions'] == 1].index,
                            y=data['short_ma'][data['positions'] == 1],
                            mode='markers', name='Buy',
                            marker=dict(color='green', symbol='triangle-up')), row=1, col=1)

figure.add_trace(go.Scatter(x=data.loc[data['positions'] == -1].index,
                            y=data['short_ma'][data['positions'] == -1],
                            mode='markers', name='Sell',
                            marker=dict(color='red', symbol='triangle-down')), row=1, col=1)

# Adding titles and axis labels

figure.update_layout(title='Moving Average Crossover Strategy for OPAP',
                     xaxis_title='Date',
                     yaxis_title='Price',
                     height=800,
                     )
figure.update_yaxes(title="Moving Averages", row=2, col=1)
figure.show()
```


Moving Average Crossover Strategy for OPAP
Stock Price
Price
15
14
13
12
11
Candlestick
Short-Term MA
Long-Term MA
Buy
Moving Averages
Moving Averages
14
13.5
13
12.5
12
11.5
11
10.5
Date
May 2022
Jul 2022
Sep 2022
Nov 2022
Jan 2023
Mar 2023

**With the Moving Average Crossover Strategy plotted, we can visually see the buy and sell signals generated by the short-term and long-term moving averages. This strategy can be used as a simple yet effective tool to help traders identify potential entry and exit points in the market**

## Summary

**Algorithmic Trading involves using pre-defined rules and algorithms to make buying and selling decisions in the financial market. These rules dictate when to buy or sell a financial instrument based on certain criteria. Algorithmic Trading is often used by institutional investors and hedge funds to execute trades quickly and efficiently.**