Exercício 3 - Algotrading

Antonio Fuziy

Lista: Exercício 3 - 16/Mai até 9h30

- Modificar a ideia 1 da aula de hoje para simular pares de até 3 ações americanas correlacionadas.
- Selecionar entre 1 e 2 anos de dados
- Precisa ajustar as quantidades para que tenham o mesmo valor financeiro. As quantidades serão inteiros múltiplos de 1.
- Entregar um **PDF** com o código e o gráfico da simulação do resultado
- Prazo: 16/Mai até 9h30 via Blackboard (Após esse prazo será considerado atrasado)
- Estritamente individual

In [1]:
```python
%matplotlib inline

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import spearmanr
from statsmodels.tsa.stattools import coint
import statsmodels.tsa.stattools as st
import time
from datetime import datetime
import random
import pandas.testing as tm
from statsmodels.tsa.stattools import adfuller
import yahoofinancials as yf

from backtesting_v2_1 import TradingSystem, MarketData, Order, Strategy, Event, evaluateHist

plt.style.use("dark_background")
```

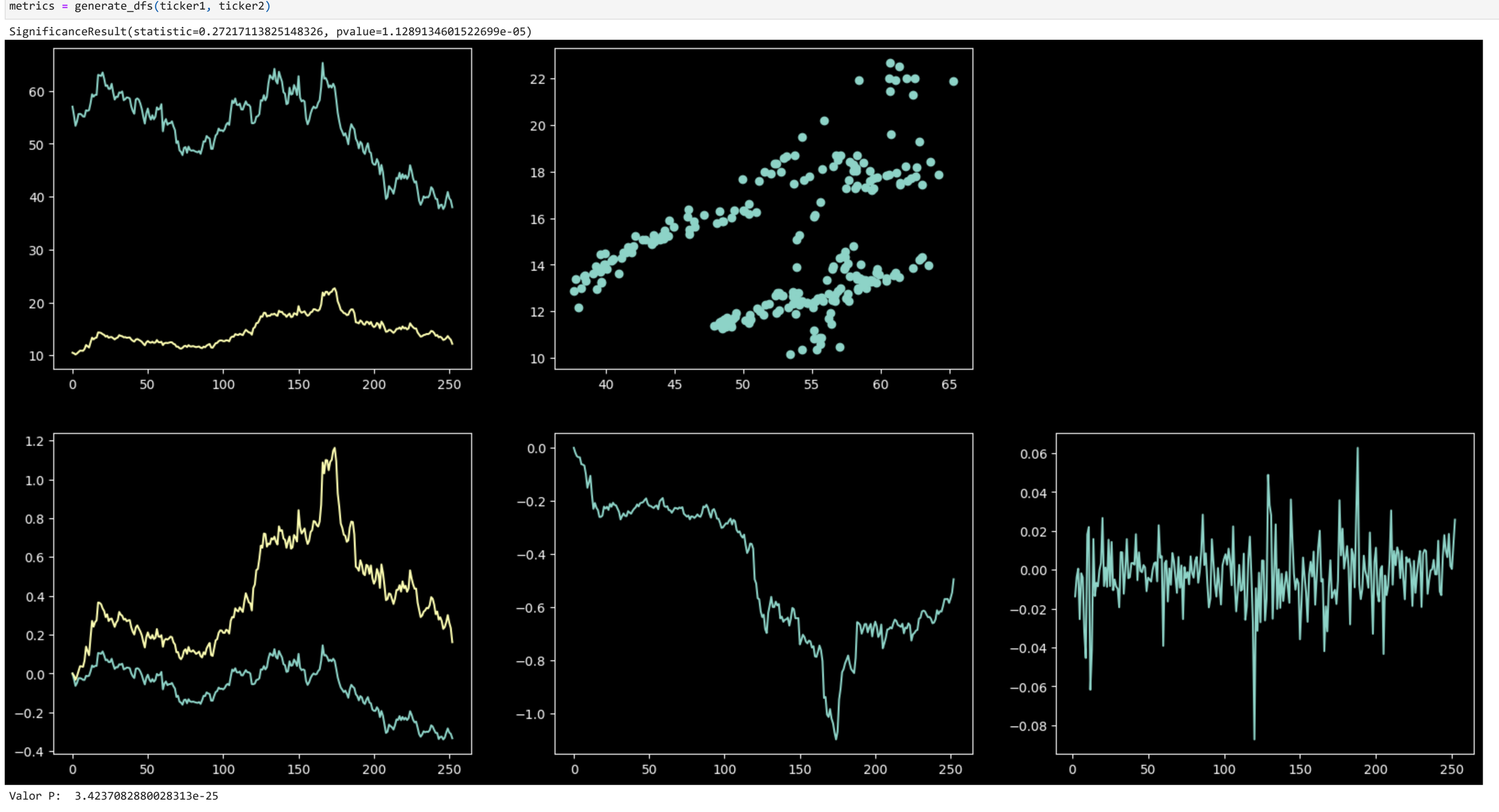
In [2]:
```python
tickers = ["GM", "F", "AZUL", "GOL", "KO", "PEP"]

start_date = '2021-05-10'
end_date =  '2022-05-10'

for ticker in tickers:
    data = yf.YahooFinancials(ticker).get_historical_price_data(start_date, end_date, 'daily')

    # Ler os dados do JSON
    raw = pd.DataFrame(data[ticker]['prices']).dropna()
    # Converter a data para o tipo correto datetime
    raw['formatted_date'] = pd.to_datetime(raw['formatted_date'])
    # Indica a data como o índice de cada linha
    raw = raw.set_index('formatted_date')
    # Removendo as colunas que não interessam
    df = raw.iloc[:,1:]

    # Acertando a ordem das colunas
    df = df.reindex(columns=['open', 'high', 'low', 'close', 'adjclose', 'volume'])
    # Salvando o CSV
    df.to_csv('{}.csv'.format(ticker))
```

In [3]:
```python
def generate_dfs(ticker1, ticker2):

    df1 = pd.read_csv(f'{ticker1}.csv')
    df2 = pd.read_csv(f'{ticker2}.csv')

    df1.set_index('formatted_date')
    df2.set_index('formatted_date')

    df_join_ = df1.join(df2, how='inner', lsuffix='_df1', rsuffix='_df2')

    plt.figure(figsize=(20, 10))

    plt.subplot(2, 3, 1)
    plt.plot(df_join_.adjclose_df1);
    plt.plot(df_join_.adjclose_df2);

    plt.subplot(2, 3, 2)
    plt.scatter(df_join_.adjclose_df1, df_join_.adjclose_df2);

    print(spearmanr(df_join_.adjclose_df1, df_join_.adjclose_df2))

    ret_df1 = df_join_.adjclose_df1/df_join_.adjclose_df1[0]-1
    ret_df2 = df_join_.adjclose_df2/df_join_.adjclose_df2[0]-1
    df_join_['diff'] = df_join_.adjclose_df1.pct_change()-df_join_.adjclose_df2.pct_change()
    df_join_ = df_join_.dropna()

    plt.subplot(2, 3, 4)
    plt.plot(ret_df1);
    plt.plot(ret_df2);

    plt.subplot(2, 3, 5)
    plt.plot(ret_df1-ret_df2);

    plt.subplot(2, 3, 6)
    df_join_['diff'] = df_join_.adjclose_df1.pct_change()-df_join_.adjclose_df2.pct_change()
    df_join_ = df_join_.dropna()
    plt.plot(df_join_['diff']);

    plt.show()

    adf, pvalue, *_ = adfuller(df_join_['diff'])
    print("Valor P: ", pvalue)

    metrics = df_join_['diff'].describe()

    df_join_diff = pd.DataFrame({'formatted_date': df_join_['formatted_date_df1'],
                        'open': df_join_['open_df1'],
                        'high': df_join_['high_df1'],
                        'low': df_join_['low_df1'],
                        'close': df_join_['close_df1'],
                        'adjclose': df_join_['diff'],
                        'volume': 0,
                        })
    df_join_diff = df_join_diff.set_index('formatted_date')

    df_join_diff.to_csv('df_join_diff.csv')

    return metrics
```

In [4]:
```python
class LongShort(Strategy):

    def __init__(self, uband, lband, ticker1, ticker2):
        self.prices = []
        self.indicator = []

        self.prices1 = []
        self.prices2 = []

        self.side = 0

        self.qty1 = 0
        self.qty2 = 0

        self.ticker1 = ticker1
        self.ticker2 = ticker2

        self.price1 = 1
        self.price2 = 1
        self.multiple = 1

        self.uband = uband
        self.lband = lband

        self.sell_signals1 = []
        self.buy_signals1 = []
        self.sell_indexes1 = []
        self.buy_indexes1 = []
        self.sell_signals2 = []
        self.buy_signals2 = []
        self.sell_indexes2 = []
        self.buy_indexes2 = []

    def receive(self, event):

        orders = []

        if event.instrument == self.ticker1:
            self.prices1.append(event.price[3])
            self.price1 = event.price[3]

        if event.instrument == self.ticker2:
            self.prices2.append(event.price[3])
            self.price2 = event.price[3]

        if event.instrument == 'df_join_diff':

            price = event.price[3] # Captura o spread

            # Montando a série de preços:
            self.prices.append(price)

            signal = self.side

            if price > self.uband:
                signal = -1 # Sell PETR3, buy PETR4
            elif price < self.lband:
                signal = 1 # Buy PETR3, sell PETR4

            if self.side != signal:
                if self.qty1 > 0:
                    self.submit(self.id, Order(self.ticker1, Order.S, self.qty1, 0))
                elif self.qty1 < 0:
                    self.submit(self.id, Order(self.ticker1, Order.B, self.qty1, 0))

                # Zeragem ticker2:
                if self.qty2 > 0:
                    self.submit(self.id, Order(self.ticker2, Order.S, self.qty2, 0))
                elif self.qty2 < 0:
                    self.submit(self.id, Order(self.ticker2, Order.B, self.qty2, 0))

                # Vamos user o mesmo tamanho porque eles tem preços semelhantes!
                # Mas não é o ideal!

                if self.price1 >= self.price2:
                    self.multiple = int(self.price1 / self.price2)
                    self.qty2 = 100 * signal * self.multiple
                    self.qty1 = -int(100 * signal)

                elif self.price1 < self.price2:
                    self.multiple = int(self.price2 / self.price1)
                    self.qty1 = 100 * signal
                    self.qty2 = -int(100 * signal) * self.multiple

                if self.qty1 > 0:
                    self.submit(self.id, Order(self.ticker1, Order.B, self.qty1, 0))
                    self.buy_signals1.append(self.price1)
                    self.buy_indexes1.append(len(self.prices1))

                elif self.qty1 < 0:
                    self.submit(self.id, Order(self.ticker1, Order.SS, self.qty1, 0))
                    self.sell_signals1.append(self.price1)
                    self.sell_indexes1.append(len(self.prices1))

                if self.qty2 > 0:
                    self.submit(self.id, Order(self.ticker2, Order.B, self.qty2, 0))
                    self.buy_signals2.append(self.price2)
                    self.buy_indexes2.append(len(self.prices2))

                elif self.qty2 < 0:
                    self.submit(self.id, Order(self.ticker2, Order.SS, self.qty2, 0))
                    self.sell_signals2.append(self.price2)
                    self.sell_indexes2.append(len(self.prices2))

                self.side = signal
        return orders

#     def fill(self, id, instrument, price, quantity, status):

#         # Imprimindo o preenchimento das ordens
#         if quantity != 0:
#             print('Fill: {0} {1}@{2}'.format(instrument, quantity, price))
```

In [5]:
```python
ticker1 = "GM"
ticker2 = "F"

metrics = generate_dfs(ticker1, ticker2)
```

```
SignificanceResult(statistic=0.27217113825148326, pvalue=1.1289134601522699e-05)
```

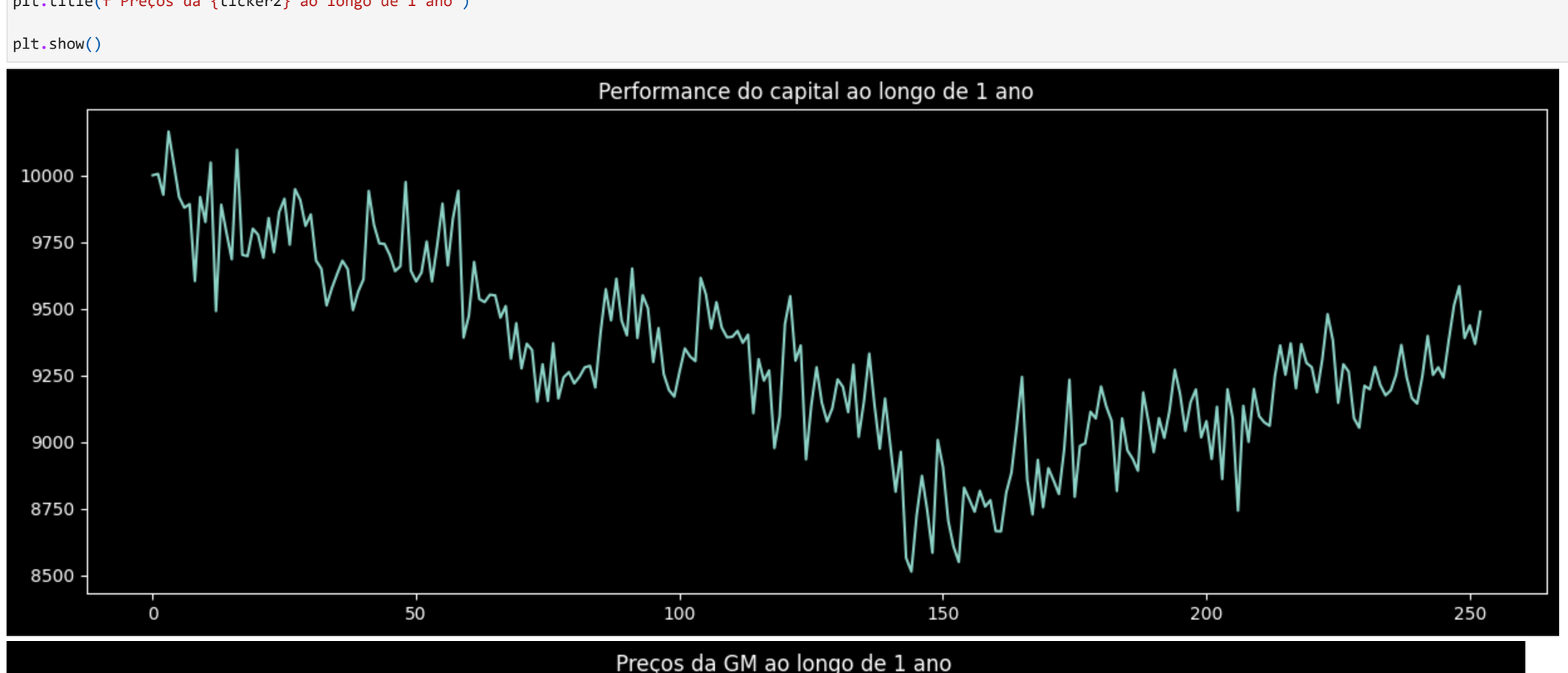


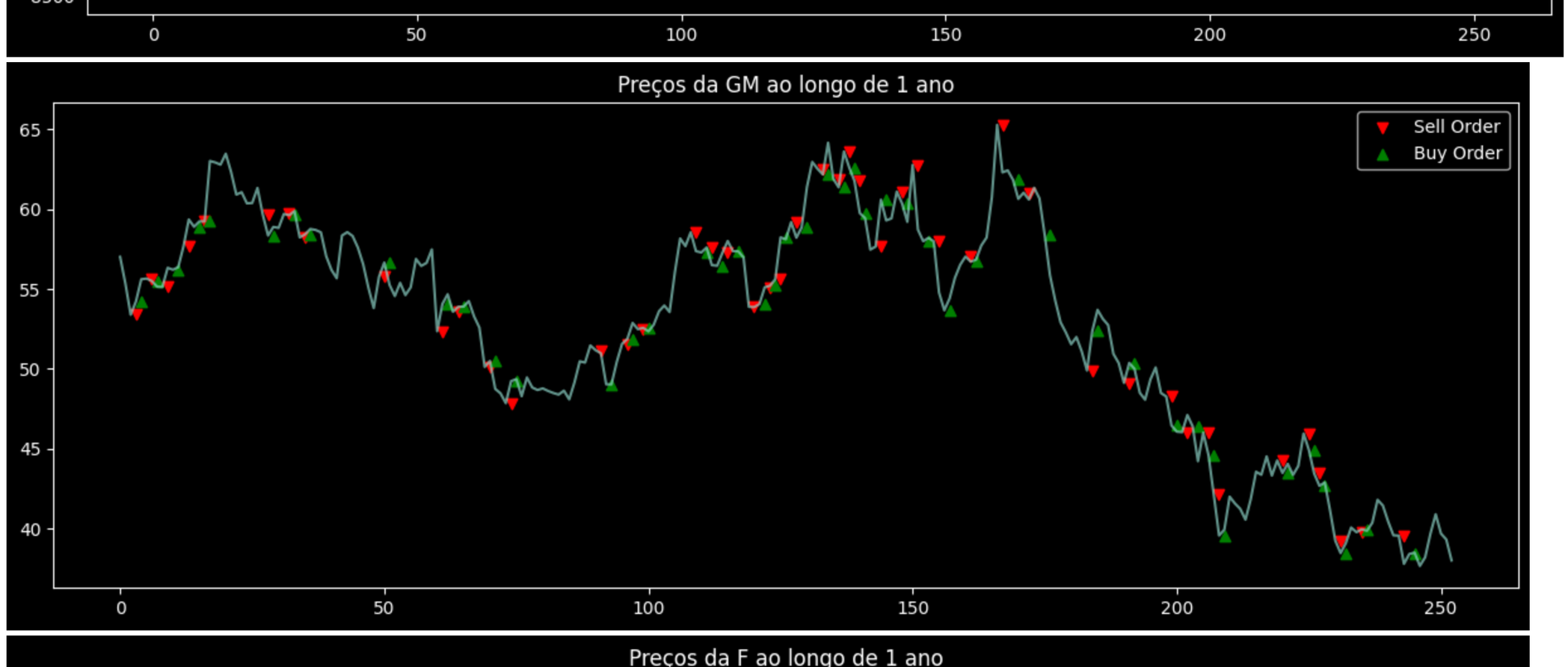
```
Valor P:  3.4237082880028313e-25
```

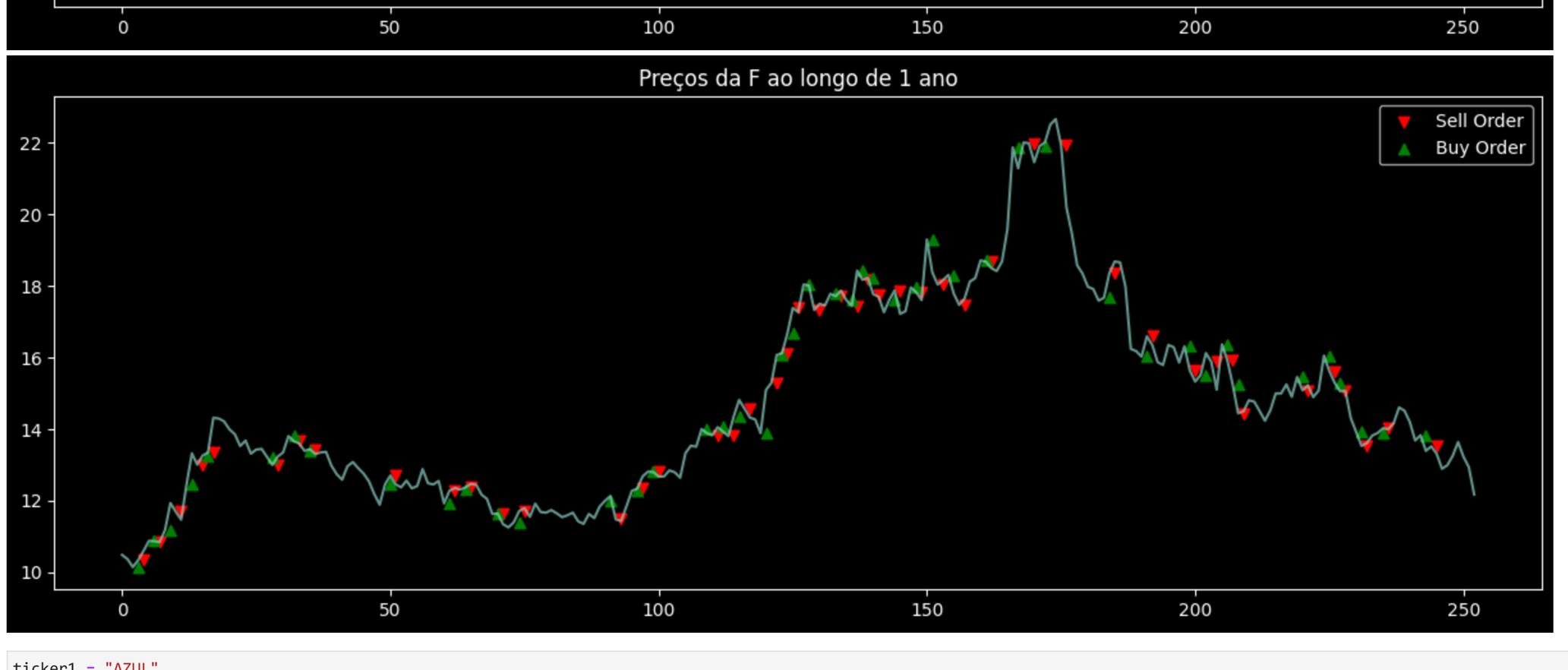
In [6]:
```python
model = LongShort(metrics[4], metrics[6], ticker1, ticker2)

print(evaluateHist(model, {ticker1: f'{ticker1}.csv', ticker2: f'{ticker2}.csv', 'df_join_diff': 'df_join_diff.csv'}))
```

```
Gross Profit: $3773.52
Gross Loss: $-4190.06
Gross Total: $-416.54

Number of trades: 92
Hitting Ratio: 50.00%
Number of profit trades: 46
Number of loss trades: 46
Average number of events per trade: 2.72

Max win trade: $404.78
Avg win trade: $82.03
Max loss trade: $-215.71
Avg loss trade: $-91.09
Avg all trades: $-4.53
Win/Loss ratio: 0.90

Max Profit: $584.50
Max Profit High/Low: $997.24
Max Drawdown: $-369.18
Max Drawdown High/Low: $-857.79

Max Allocation: $6529.26
Avg Allocation: $5381.70
Max Cash Required (margin): $6529.26

Gross Total: $-416.54
Total Fees: $36.80
Total Taxes: $575.16
Net Total: $-1028.50

Gross Return: -4.70%
Average Return: -0.05%
Net Return: -16.36%
Net Return Avg Alocation: -19.11%

Number of days: 253
Initial Capital: $10000.00
Risk Free Rate: 13.75% yearly/0.0511% daily
Total Carry: $521.86
Net Total + Carry: $-506.64
Net Return Capital: -5.07%
Net Return Capital Yearly: -5.05%
```

In [7]:
```python
plt.figure(figsize=(15,5))
plt.plot([day[0] for day in model._days.values()])
plt.title("Performance do capital ao longo de 1 ano")
plt.show()

plt.figure(figsize=(15,5))
plt.plot(model.prices1, alpha=0.7)
plt.scatter(model.sell_indexes1, model.sell_signals1, marker="v", color="red", label="Sell Order")
plt.scatter(model.buy_indexes1, model.buy_signals1, marker="^", color="green", label="Buy Order")
plt.legend()
plt.title(f"Preços da {ticker1} ao longo de 1 ano")
plt.show()

plt.figure(figsize=(15,5))
plt.plot(model.prices2, alpha=0.7)
plt.scatter(model.sell_indexes2, model.sell_signals2, marker="v", color="red", label="Sell Order")
plt.scatter(model.buy_indexes2, model.buy_signals2, marker="^", color="green", label="Buy Order")
plt.legend()
plt.title(f"Preços da {ticker2} ao longo de 1 ano")

plt.show()
```

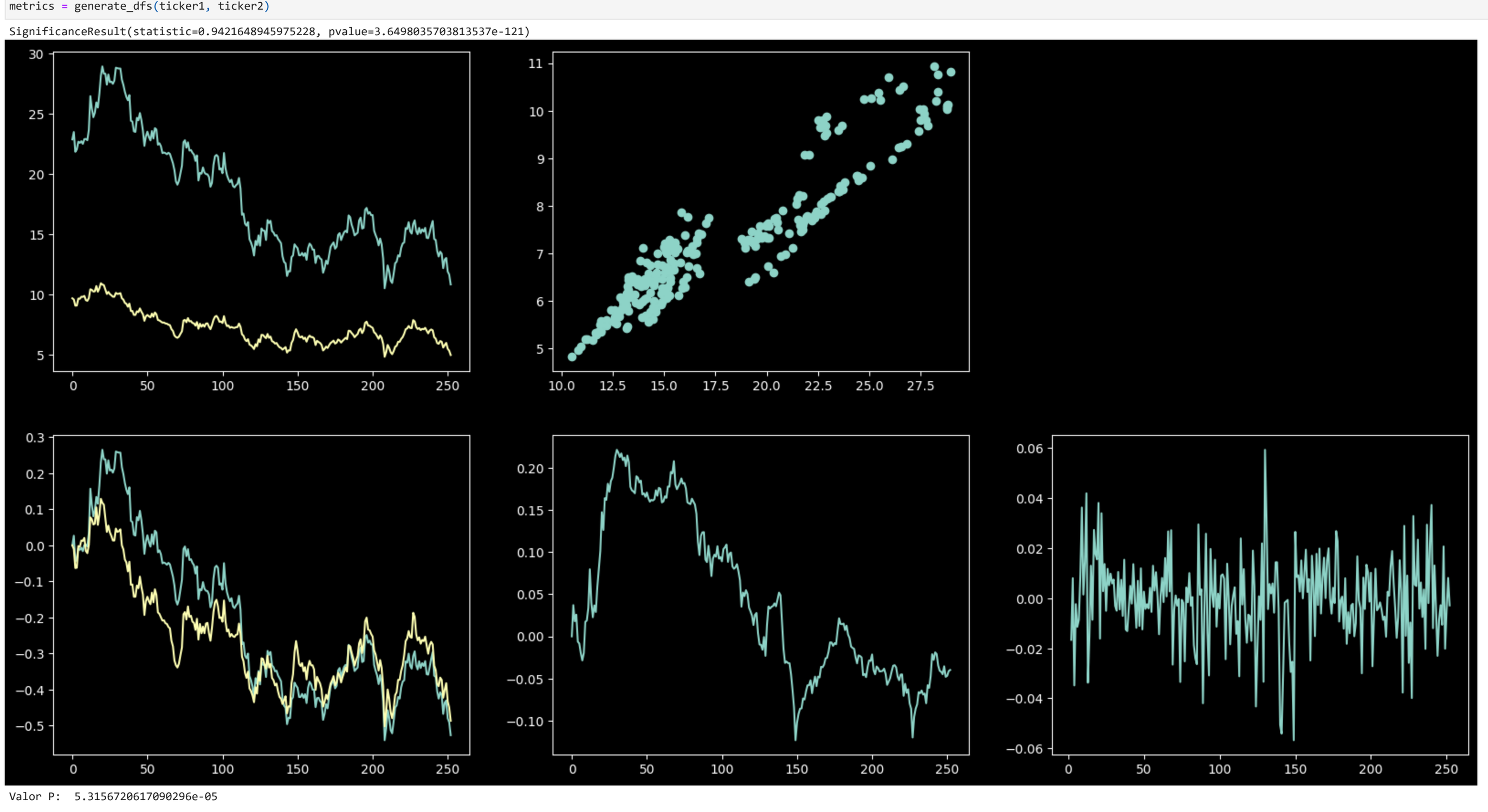






In [8]:
```python
ticker1 = "AZUL"
ticker2 = "GOL"

metrics = generate_dfs(ticker1, ticker2)
```

```
SignificanceResult(statistic=0.9421648945975228, pvalue=3.6498035703813537e-121)
```

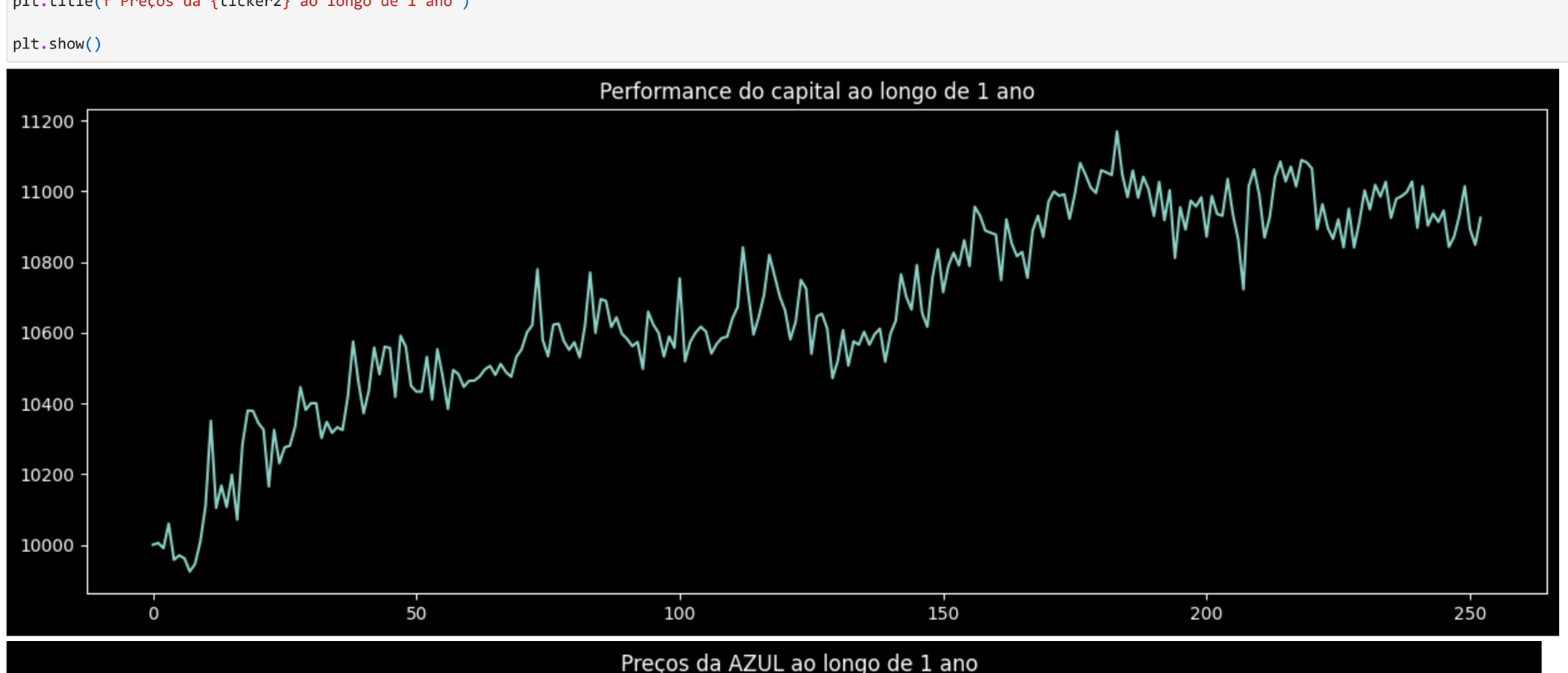


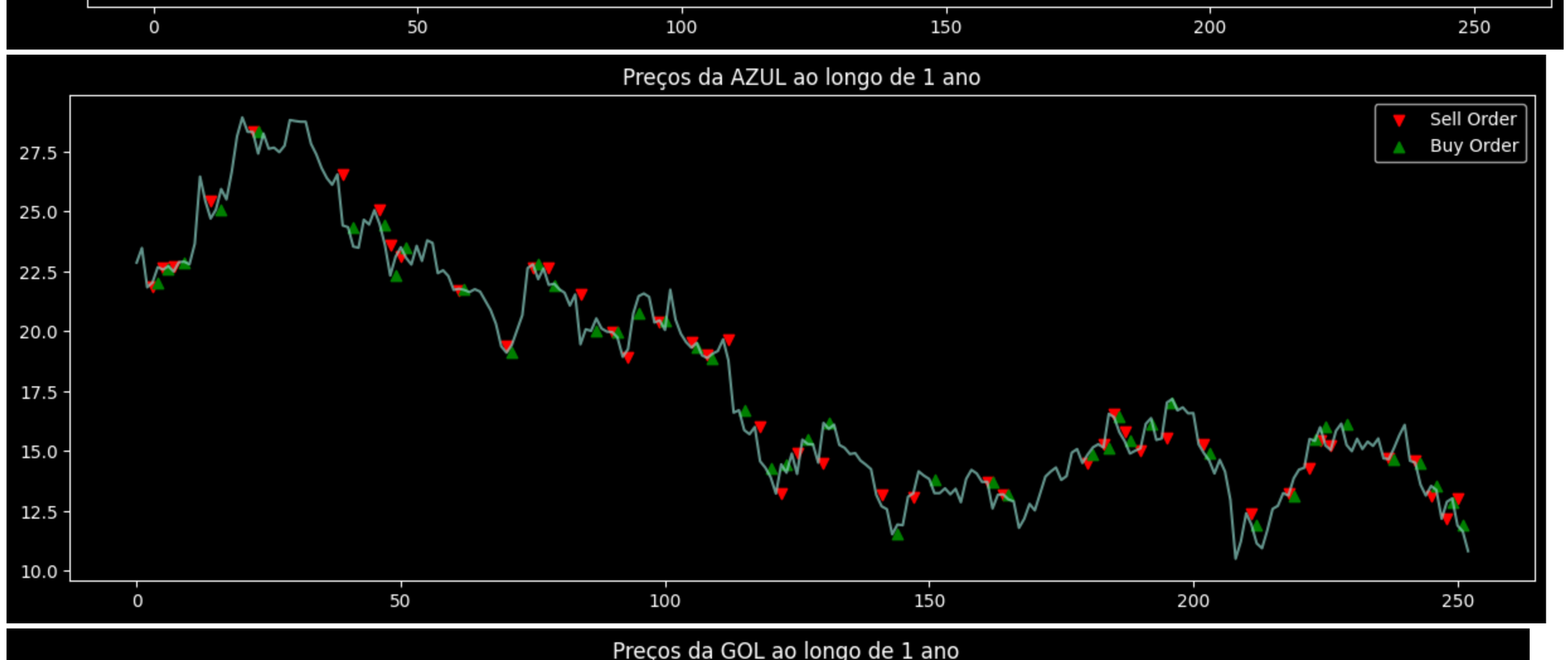
```
Valor P:  5.315672061709029e-05
```

In [9]:
```python
model = LongShort(metrics[4], metrics[6], ticker1, ticker2)

print(evaluateHist(model, {ticker1: f'{ticker1}.csv', ticker2: f'{ticker2}.csv', 'df_join_diff': 'df_join_diff.csv'}))
```

```
Gross Profit: $1574.00
Gross Loss: $-1513.00
Gross Total: $61.00

Number of trades: 90
Hitting Ratio: 40.00%
Number of profit trades: 36
Number of loss trades: 53
Average number of events per trade: 2.78

Max win trade: $223.00
Avg win trade: $43.72
Max loss trade: $-96.00
Avg loss trade: $-28.55
Avg all trades: $0.68
Win/Loss ratio: 1.53

Max Profit: $395.00
Max Profit High/Low: $367.00
Max Drawdown: $-269.00
Max Drawdown High/Low: $-267.50

Max Allocation: $2833.00
Avg Allocation: $1783.21
Max Cash Required (margin): $2833.00

Gross Total: $61.00
Total Fees: $36.00
Total Taxes: $238.98
Net Total: $-213.98

Gross Return: -2.52%
Average Return: -0.03%
Net Return: -18.03%
Net Return Avg Alocation: -12.00%

Number of days: 253
Initial Capital: $10000.00
Risk Free Rate: 13.75% yearly/0.0511% daily
Total Carry: $1144.69
Net Total + Carry: $930.71
Net Return Capital: 9.31%
Net Return Capital Yearly: 9.27%
```

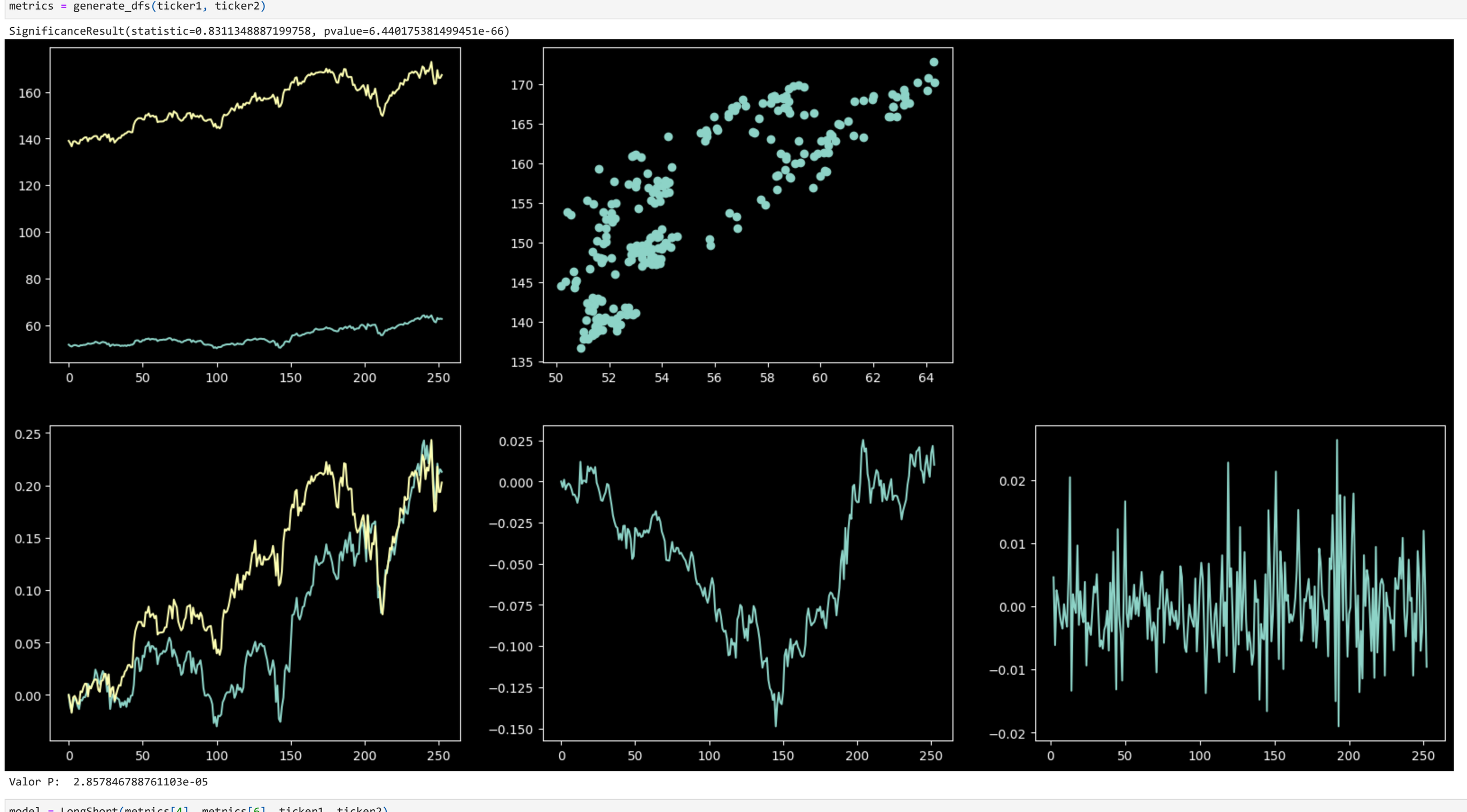
In [10]:
```python
plt.figure(figsize=(15,5))
plt.plot([day[0] for day in model._days.values()])
plt.title("Performance do capital ao longo de 1 ano")
plt.show()

plt.figure(figsize=(15,5))
plt.plot(model.prices1, alpha=0.7)
plt.scatter(model.sell_indexes1, model.sell_signals1, marker="v", color="red", label="Sell Order")
plt.scatter(model.buy_indexes1, model.buy_signals1, marker="^", color="green", label="Buy Order")
plt.legend()
plt.title(f"Preços da {ticker1} ao longo de 1 ano")
plt.show()

plt.figure(figsize=(15,5))
plt.plot(model.prices2, alpha=0.7)
plt.scatter(model.sell_indexes2, model.sell_signals2, marker="v", color="red", label="Sell Order")
plt.scatter(model.buy_indexes2, model.buy_signals2, marker="^", color="green", label="Buy Order")
plt.legend()
plt.title(f"Preços da {ticker2} ao longo de 1 ano")

plt.show()
```





In [11]:
```python
ticker1 = "KO"
ticker2 = "PEP"

metrics = generate_dfs(ticker1, ticker2)
```

```
SignificanceResult(statistic=0.8311348887199758, pvalue=6.440175381499451e-66)
```

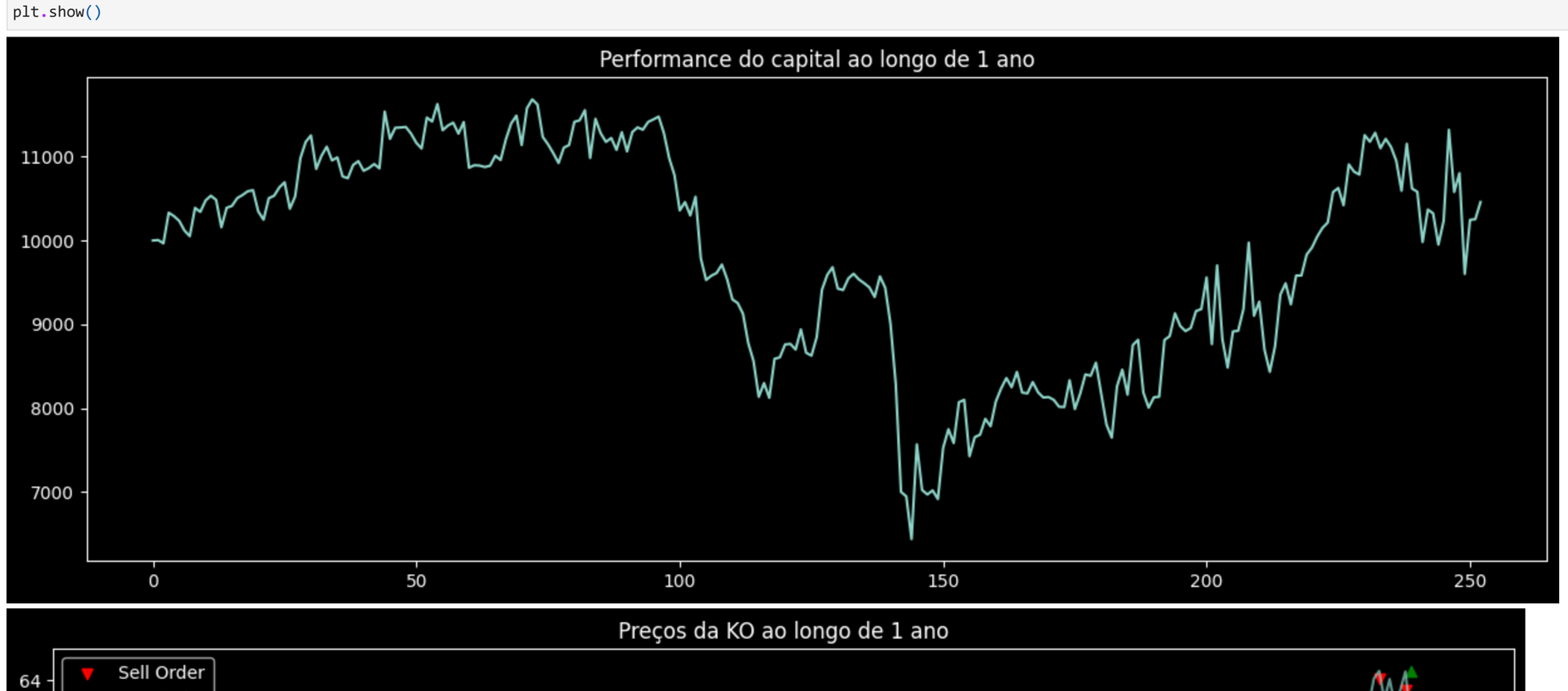


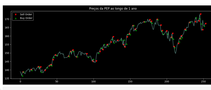
```
Valor P:  2.857846788761103e-05
```

In [12]:
```python
model = LongShort(metrics[4], metrics[6], ticker1, ticker2)

print(evaluateHist(model, {ticker1: f'{ticker1}.csv', ticker2: f'{ticker2}.csv', 'df_join_diff': 'df_join_diff.csv'}))
```

```
Gross Profit: $17596.11
Gross Loss: $-12334.71
Gross Total: $5261.41

Number of trades: 96
Hitting Ratio: 61.46%
Number of profit trades: 59
Number of loss trades: 36
Average number of events per trade: 2.60

Max win trade: $1218.13
Avg win trade: $298.24
Max loss trade: $-1092.06
Avg loss trade: $-342.63
Avg all trades: $54.81
Win/Loss ratio: 0.87

Max Profit: $1218.13
Max Profit High/Low: $2730.07
Max Drawdown: $-1092.06
Max Drawdown High/Low: $-2843.16

Max Allocation: $48575.17
Avg Allocation: $32511.97
Max Cash Required (margin): $48575.17

Gross Total: $5261.41
Total Fees: $38.40
Total Taxes: $2675.85
Net Total: $2547.16

Gross Return: 16.82%
Average Return: 0.18%
Net Return: 8.57%
Net Return Avg Alocation: 7.83%

Number of days: 253
Initial Capital: $10000.00
Risk Free Rate: 13.75% yearly/0.0511% daily
Total Carry: $-2083.88
Net Total + Carry: $463.28
Net Return Capital: 4.63%
Net Return Capital Yearly: 4.61%
```

In [13]:
```python
plt.figure(figsize=(15,5))
plt.plot([day[0] for day in model._days.values()])
plt.title("Performance do capital ao longo de 1 ano")
plt.show()

plt.figure(figsize=(15,5))
plt.plot(model.prices1, alpha=0.7)
plt.scatter(model.sell_indexes1, model.sell_signals1, marker="v", color="red", label="Sell Order")
plt.scatter(model.buy_indexes1, model.buy_signals1, marker="^", color="green", label="Buy Order")
plt.legend()
plt.title(f"Preços da {ticker1} ao longo de 1 ano")
plt.show()

plt.figure(figsize=(15,5))
plt.plot(model.prices2, alpha=0.7)
plt.scatter(model.sell_indexes2, model.sell_signals2, marker="v", color="red", label="Sell Order")
plt.scatter(model.buy_indexes2, model.buy_signals2, marker="^", color="green", label="Buy Order")
plt.legend()
plt.title(f"Preços da {ticker2} ao longo de 1 ano")

plt.show()
```




Prezzo da PAC ai tempi de 'I Lyes
Sell Order
Buy Order