In [1]:

```
#Disclaimer: The relative arbitrage strategy was
#not fully implemented until October, 2022.
#Prior to October, 2022, it was a mixture of mostly
#put spread and a few ITM call as well as futures
#for quick delta adjustment.
#Since then, this relative arbitrage strategy has
#been fully and consistenly implemented.
```

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:

```
# Load the Excel file
excel_file = pd.ExcelFile('E:\Derivatives Trading\TAIEX derivatives trading record

# Get the sheet you want to read
sheet_name = 'ForPython' # Replace with the name of the sheet you want to read
df = excel_file.parse(sheet_name)
```

In [3]:

```
# Output data information
print(df.head())
```

```
        Date   PnL Index     TAIEX    VIX   Returns  Unnamed: 5  Unnamed: 6  \
0 2022-07-01  100.000000  14343.08  27.01  0.000000         NaN         NaN   
1 2022-07-04   95.577858  14217.06  27.56 -0.044221         NaN         NaN   
2 2022-07-05   93.953178  14349.20  27.18 -0.016998         NaN         NaN   
3 2022-07-06   92.057052  13985.51  29.40 -0.020182         NaN         NaN   
4 2022-07-07   92.698962  14335.27  28.26  0.006973         NaN         NaN   

    Base  
0  100.0  
1    NaN  
2    NaN  
3    NaN  
4    NaN  
```

In [4]:

```
#******Plotting setup*****#
# Generate some data
Date = df["Date"]
Date
y1 =df["PnL Index"]
y1
y2 = df["TAIEX"]
y2
```

Out[4]:

```
0      14343.08
1      14217.06
2      14349.20
3      13985.51
4      14335.27
         ...   
246    16652.80
247    16898.91
248    16962.03
249    17061.40
250    17283.71
Name: TAIEX, Length: 251, dtype: float64
```

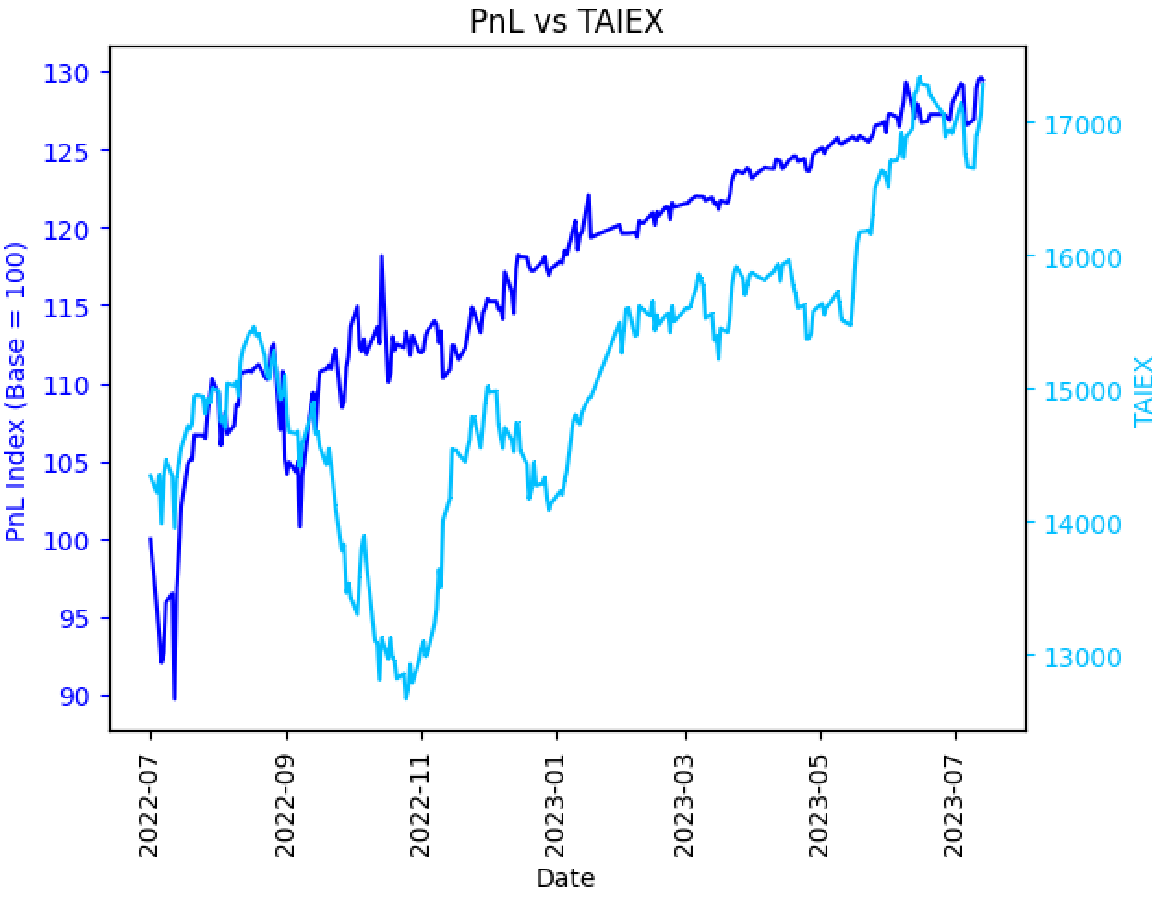
In [5]:

```
# Create the plot and set the first y-axis (left)
fig, ax1 = plt.subplots()
plt.xticks(rotation=90)
ax1.plot(Date, y1, 'b-')
```

```
ax1.set_xlabel('Date')
ax1.set_ylabel('PnL Index (Base = 100)', color='b')
ax1.tick_params('y', colors='b')

# Set the second y-axis (right)
ax2 = ax1.twinx()
ax2.plot(Date, y2, color='deepskyblue', marker=',')
ax2.set_ylabel('TAIEX', color='deepskyblue')
ax2.tick_params('y', colors='deepskyblue')

# Show the plot
plt.title('PnL vs TAIEX')
plt.show()
```



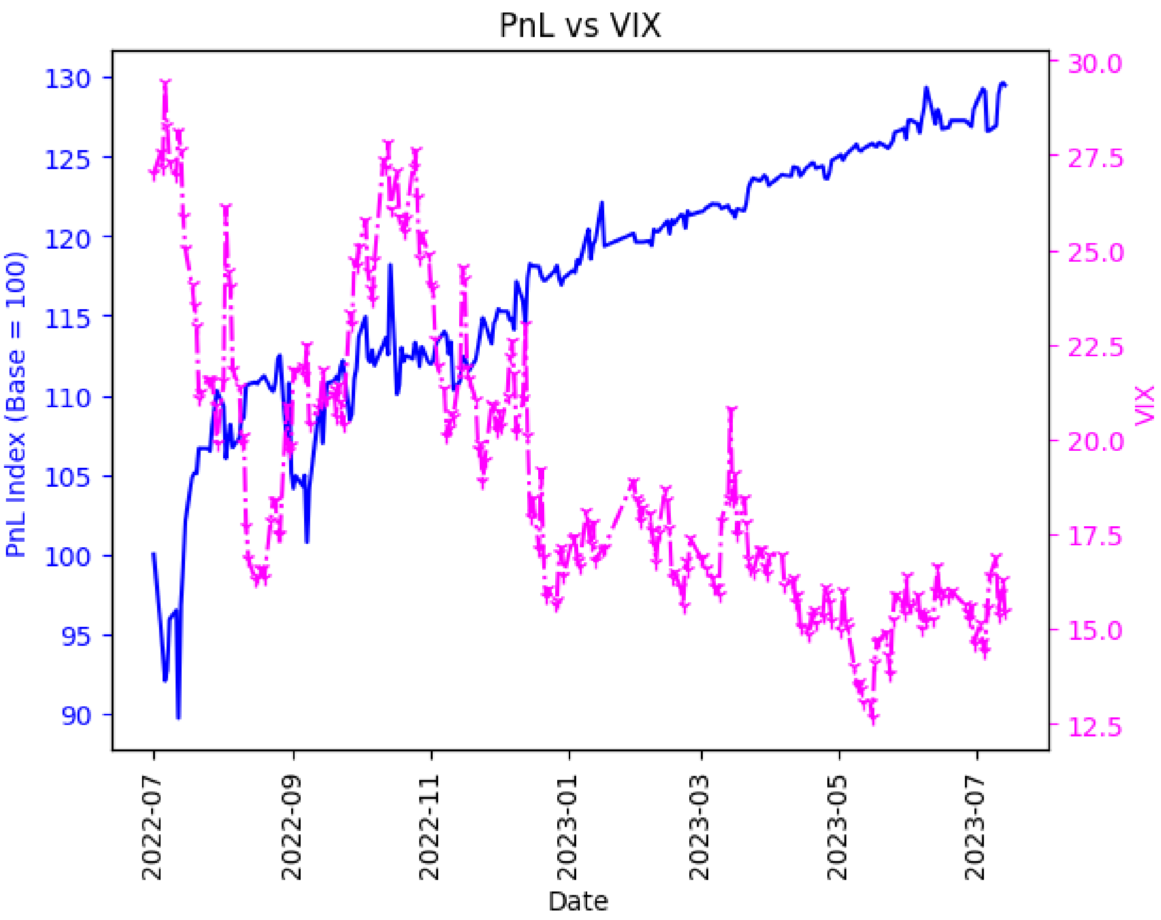
In [6]:

```
#Pnl vs VIX
y3 = df["VIX"]
y3

# Create the plot and set the first y-axis (left)
fig, ax1 = plt.subplots()
plt.xticks(rotation=90)
ax1.plot(Date, y1, 'b-')
ax1.set_xlabel('Date')
ax1.set_ylabel('PnL Index (Base = 100)', color='b')
ax1.tick_params('y', colors='b')

# Set the second y-axis (right)
ax3 = ax1.twinx()
ax3.plot(Date, y3, 'fuchsia', marker='1', linestyle='-.')
ax3.set_ylabel('VIX', color='fuchsia')
ax3.tick_params('y', colors='fuchsia')

# Show the plot
plt.title('PnL vs VIX')
plt.show()
```



In [7]:

```
#Sharpe ratio
# Read in the portfolio returns data from a CSV file
R_first=df["PnL Index"].iloc[0,]
R_first
R_last=df["PnL Index"].iloc[165,]  #Always excel's actual row-2
R_last
```

Out[7]:

```
121.98400800102736
```

In [8]:

```
portfolio_returns=(R_last-R_first)/R_first
portfolio_returns
```

Out[8]:

```
0.21984008001027364
```

In [9]:

```
daily_returns=df["Returns"]
daily_returns
```

Out[9]:

```
0       0.000000
1      -0.044221
2      -0.016998
3      -0.020182
4       0.006973
          ...
246     0.002461
247     0.015662
248     0.004346
249     0.001079
250    -0.001214
Name: Returns, Length: 251, dtype: float64
```

In [10]:

```
# Max Drawdown Calculation for PnL Index
cumulative_returns = (1 + df["Returns"]).cumprod()
```

```
cumulative_max = cumulative_returns.cummax()
drawdown = (cumulative_returns / cumulative_max) - 1
max_drawdown = drawdown.min()

print("Max Drawdown:", max_drawdown)
```

```
Max Drawdown: -0.10420949154156467
```

In [11]:

```
# Calculate the Profit Factor
positive_returns = daily_returns[daily_returns > 0].sum()
negative_returns = daily_returns[daily_returns < 0].sum()

# Avoid division by zero
if negative_returns != 0:
    profit_factor = abs(positive_returns / negative_returns)
else:
    profit_factor = float('inf')

print("Profit Factor:", profit_factor)
```

```
Profit Factor: 1.330561761394395
```

In [12]:

```
# Calculate the excess returns and standard deviation
risk_free_rate = 0.0145  # Taiwan savings rate
excess_returns = portfolio_returns - risk_free_rate
std_dev = np.std(daily_returns)
print("Standard Deviation of Daily Return:", std_dev)
```

```
Standard Deviation of Daily Return: 0.014161247575183108
```

In [15]:

```
# Calculate the Sharpe ratio
Sharpe_Ratio = excess_returns / std_dev
print("Sharpe Ratio:", Sharpe_Ratio)
```

```
Sharpe Ratio: 14.500140536355147
```

In [16]:

```
#Annualized Sharpe ratio
Annualized_Sharpe_Ratio=Sharpe_Ratio*np.sqrt(250)
print("Annualized Sharpe Ratio:", Annualized_Sharpe_Ratio)
```

```
Annualized Sharpe Ratio: 229.2673524370891
```

In [ ]:

In [ ]:

In [ ]: