Python Project Documentation

Project Name: Stock ETF BUY/ SELL Recommendation & Price Trend Visualization System
Course: Python Programming Lab
Student: Amit Thakur
Institute: IIIT Bhopal
Scholar Number: 25P03F0040
Coding_File:https://github.com/4athakur/Python-Project/blob/main/NSE_ETF_Slippage_project.py

---

1. Introduction

This is my python project which help, many small investors buy **ETF (Exchange Traded Fund)** like NIFTYBEES, BANKBEES etc. But most of the time they don't check **how much difference is there between actual INAV and market price**. This difference is called **slippage or premium/discount**.

In this project I have created a **Python based small tool** which:

- Fetch live ETF price from NSE using `nsepython`
- Calculate **slippage percentage** (CMP vs iNAV)
- Prepare a small recommendation style table
- Optionally send this table directly on **WhatsApp**
- Also show **last few days price trend** for each ETF using `yfinance` and `matplotlib`.

---

2. Objective

Main objectives of this project are:

1. To practice **real world data fetching** using APIs / Python packages.
2. To calculate **slippage difference** between ETF **Current Market Price (CMP)** and **iNAV**.

3. To display the result in a **readable table using pandas DataFrame**.
4. To send the table on **WhatsApp automatically** for daily stock recommedation.
5. To analyse **short term trend** of selected ETFs using historical data and graphs.

---

3. Tools & Technologies Used

- **Programming Language:** Python
- **Libraries:**
  - `nsepython` – It fetch live data from NSE
  - `pandas` – This library create and handle table like data
  - `yfinance` – this fetch historical stock / ETF price data
  - `matplotlib` – It helps to plot line charts for price trend
  - `pywhatkit` – This library help us to send message on WhatsApp
- **Platform:** Any system with Python latest version, visual studio code, whatsapp, A Browser

---

4. Project Overview

The project mainly works in **two parts**:

1. **Slippage Calculation & WhatsApp Recommedation**
   - Take a fixed list of ETF symbols.
   - For each symbol, fetch **last price (CMP)** and **iNAV** from NSE.
   - Calculate **percentage diff** between them.
   - Store the result in a pandas DataFrame.
   - Ask user if he wants today's stock recommedation.

- If yes, send the DataFrame as text message on WhatsApp.

2. **Historical Trend Analysis**
    - Ask user how many days of trend he wants to see.
    - Use `yfinance` to download last N days data for each ETF.
    - Plot a **line graph of Close price vs Date** using matplotlib.
    - Show graph one by one for all ETFs.

---

5. Execution Flow (Logic)

1. Define a list of **official ETF symbols** (like NIFTYBEES, BANKBEES etc.).
2. For each symbol:
    - Call `nse_eq(symbol)` from `nsepython`.
    - Extract `lastPrice` and `iNavValue` from `priceInfo`.
    - Calculate slippage:
      `diff_percentage = (CMP - iNAV) / CMP * 100`
    - Save result in a dictionary.
3. Convert dictionary to **pandas DataFrame** for better display.
4. Ask user:
   `"Would you like to get Today's Stock recomedation y\n"`
    - If user enters `y`, send DataFrame via WhatsApp using `pywhatkit.sendwhatmsg_instantly()`.
5. Ask user again if he wants **price trend graphs**.
6. If yes:
    - Ask for number of days.
    - For each symbol:

- Use `yfinance.download()` to get history.
- Plot Date vs Close Price.
- Show graph using `plt.show()`.

---

6. Code Snippets with Explanation

6.1 Importing Libraries & Configuration

```
from nsepython import nse_eq, nse_eq_symbols
import pywhatkit
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt

phone_number = "+918105749018"  # replace with your own WhatsApp
number
```

- `nse_eq` is used to get data of particular NSE symbol.
- `phone_number` is the number where WhatsApp message will be sent.

6.2 Defining ETF List and Fetching CMP

```
official_symbol =[
    "NIFTYBEES",
    "BANKBEES",
    "JUNIORBEES",
    "GOLDBEES",
    "METALIETF",
    "SILVERBEES",
    "LIQUIDCASE",
    "ITBEES",
    "PHARMABEES",
    "PSUBNKBEES"
]

final = []
stock_name = {}
```

```
finallll = {}

for ii, i in enumerate(official_symbol):
    data = nse_eq(i)
    price_info = data.get('priceInfo', {})
    last_price = price_info.get('lastPrice')

    if last_price is None:
        print(f"{i}: No lastPrice available, skipping")
        continue

    final.append(last_price)
    stock_name.update({data['info']['companyName']: {i:
last_price}})
    finallll.update({i: last_price})
```

In Above code:

- Loops through each ETF symbol.
- Fetches `lastPrice` for each.
- Stores results in different dictionaries for later use.

6.3 Calculating Slippage Difference

```
difference = {}
for i in official_symbol:
    data = nse_eq(i)
    price_info = data.get('priceInfo', {})
    current_market_price = price_info.get('lastPrice')
    inav_price = price_info.get('iNavValue')

    if current_market_price is None or inav_price is None:
        print(f"{i}: Missing price data, skipping")
        continue

    inav_price = float(inav_price)
    diff = current_market_price - inav_price
    diff_percentage = (diff / current_market_price) * 100
    difference.update({i: diff_percentage})
    print(f"{i}: CMP={current_market_price}, iNAV={inav_price},
Diff%={diff_percentage:.2f}")

info = pd.DataFrame(list(difference.items()), columns=["Stock",
"Slippage Difference"])
```

Here:

- For each symbol we again fetch data.
- We take both **CMP (`lastPrice`)** and **iNAV (`iNavValue`)**.
- Then calculate percentage difference and store in `difference` dictionary.
- Finally convert to DataFrame called `info`.

6.4 Sending WhatsApp Recommedation

```
ask = input("Would you like to get Today's Stock recommedation
y\\n")
if(ask == 'y'):
    pywhatkit.sendwhatmsg_instantly(phone_number, info.to_string())
```

- Asks user if they want today’s recommedation.
- If yes, `pywhatkit` sends the full DataFrame as plain text message.
- This makes it easy to just open WhatsApp and see the slippage table.

6.5 Second Feature Price Trend Analysis

```
DAYS = 10  # how many days of history to fetch
def fetch_and_plot_stocks(symbols, days=DAYS):
    for sym in symbols:
        ticker = f"{sym}.NS"
        print(f"\nFetching {days}-day data for {sym}...")
        try:
            df = yf.download(
                ticker,
                period=f"{days}d",
                interval="1d",
                auto_adjust=False,
                progress=False
            )

            if df.empty:
                print(f"[WARN] No data for {sym}. Skipping.")
                continue

            df = df.reset_index()

            df.plot(
                x="Date",
                y="Close",
                title=f"{sym} - Last {days} Days Price Trend",
                ylabel="Price (₹)",
                xlabel="Date",
                legend=False,
                grid=True,
                figsize=(10, 5)
            )
            plt.show()
        except Exception as e:
            print(f"[ERROR] Could not fetch data for {sym}: {e}")
```

- `fetch_and_plot_stocks` takes list of symbols and number of days.
- For each symbol it calls `yfinance.download()` to get data.
- Then uses pandas built-in plot to show **Date vs Close price**.

6.6 Working of Trend Analysis

```
ask2 = input("\nWould you like to get Today's Stock recomedation
y\\n")
if(ask2 == 'y'):
    if __name__ == "__main__":
        DAYS = input("How Many days of price trend would you like to
see ?")
        print("Starting to fetch and display 30-day stock price
trends...\n")
        fetch_and_plot_stocks(official_symbol, DAYS)
        print("\nAll available stock graphs displayed
successfully.")
```

- Again asks user if they want to see stock recommedation + trend.
- Then asks how many days of trend.
- Calls `fetch_and_plot_stocks` with official symbols and days.
- Finally prints message after all graphs are shown.

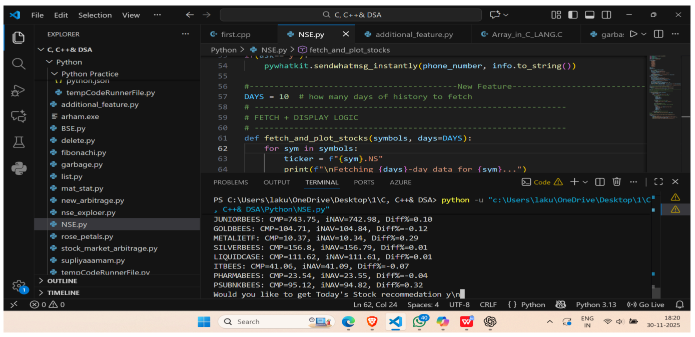
(Note: here `DAYS` from `input()` is string, ideally it should be converted to `int`, but for documentation I kept code as it is.)

---

7. Sample Output

Below are some sample outputs of the project

7.1 Output for Slippage Calculation

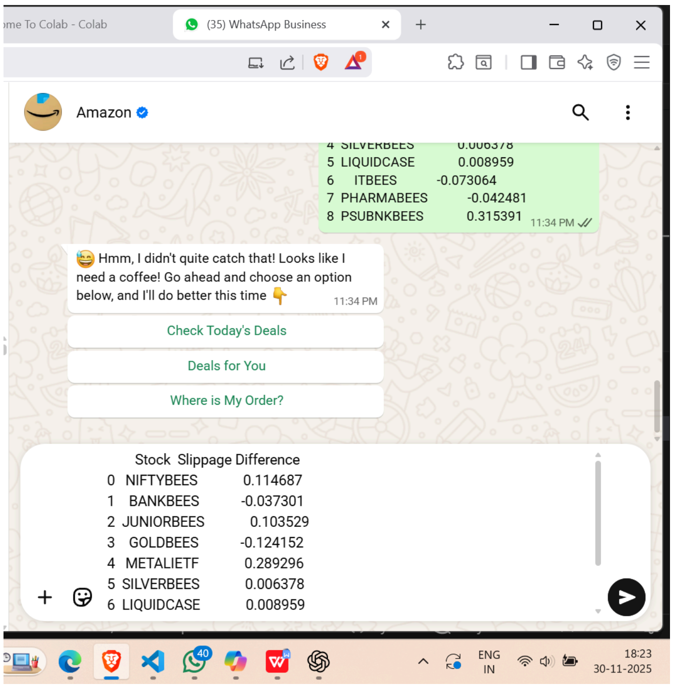


```
        pywhatkit.sendwhatmsg_instantly(phone_number, info.to_string())

#---------------------------------------New Feature-----------------------------
DAYS = 10  # how many days of history to fetch
# ---------------------------------------------------------------
# FETCH + DISPLAY LOGIC
# ---------------------------------------------------------------
def fetch_and_plot_stocks(symbols, days=DAYS):
    for sym in symbols:
        ticker = f"{sym}.NS"
        print(f"\nFetching {days}-day data for {sym}...")
```

```
PS C:\Users\laku\OneDrive\Desktop\1\C, C++& DSA> python -u "c:\Users\laku\OneDrive\Desktop\1\C
, C++& DSA\Python\NSE.py"
JUNIORBEES: CMP=743.75, iNAV=742.98, Diff%=0.10
GOLDBEES: CMP=104.71, iNAV=104.84, Diff%=-0.12
METALIETF: CMP=10.37, iNAV=10.34, Diff%=0.29
SILVERBEES: CMP=156.8, iNAV=156.79, Diff%=0.01
LIQUIDCASE: CMP=111.62, iNAV=111.61, Diff%=0.01
ITBEES: CMP=41.06, iNAV=41.09, Diff%=-0.07
PHARMABEES: CMP=23.54, iNAV=23.55, Diff%=-0.04
PSUBNKBEES: CMP=95.12, iNAV=94.82, Diff%=0.32
Would you like to get Today's Stock recommedation y\n
```

7.2 DataFrame Sent to WhatsApp



```
4  SILVERBEES        0.006378
5  LIQUIDCASE        0.008959
6     ITBEES        -0.073064
7  PHARMABEES        -0.042481
8  PSUBNKBEES        0.315391   11:34 PM
```

😂 Hmm, I didn't quite catch that! Looks like I need a coffee! Go ahead and choose an option below, and I'll do better this time 👇 11:34 PM

```
         Stock  Slippage Difference
0   NIFTYBEES        0.114687
1    BANKBEES       -0.037301
2  JUNIORBEES        0.103529
3    GOLDBEES       -0.124152
4   METALIETF        0.289296
5  SILVERBEES        0.006378
6  LIQUIDCASE        0.008959
```

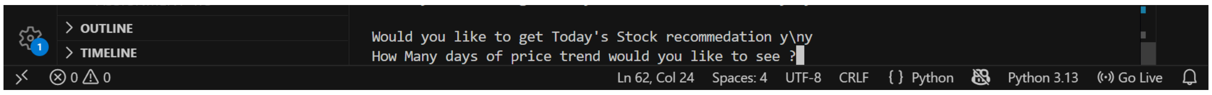
This exact table is also sent as **WhatsApp message** if user enters y.

7.3 Example User Input Flow

```
Would you like to get Today's Stock recommedation y\n y
[WhatsApp message sent]

Would you like to get Today's Stock recommedation y\n y

How Many days of price trend would you like to see ?
```



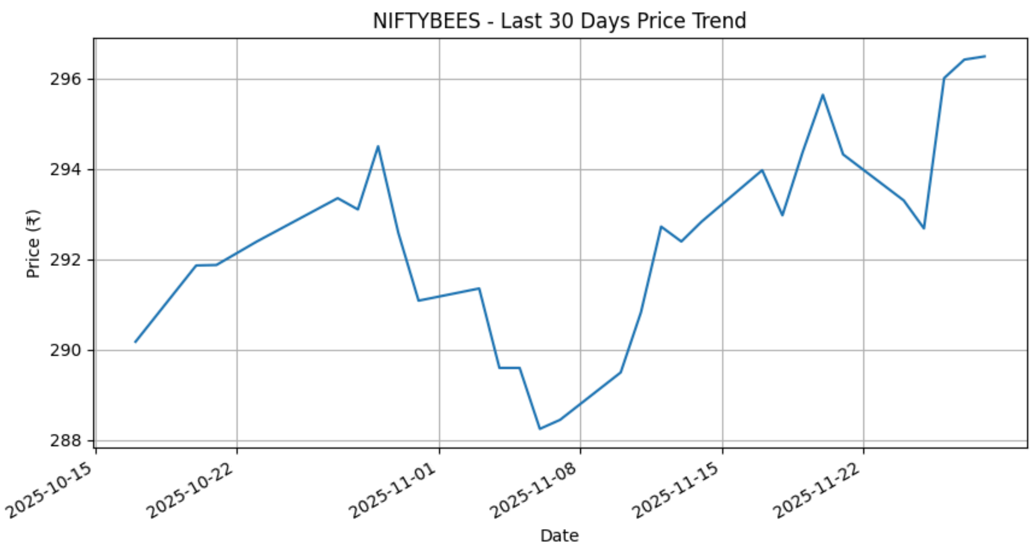
```
Would you like to get Today's Stock recommedation y\ny
How Many days of price trend would you like to see ?
```

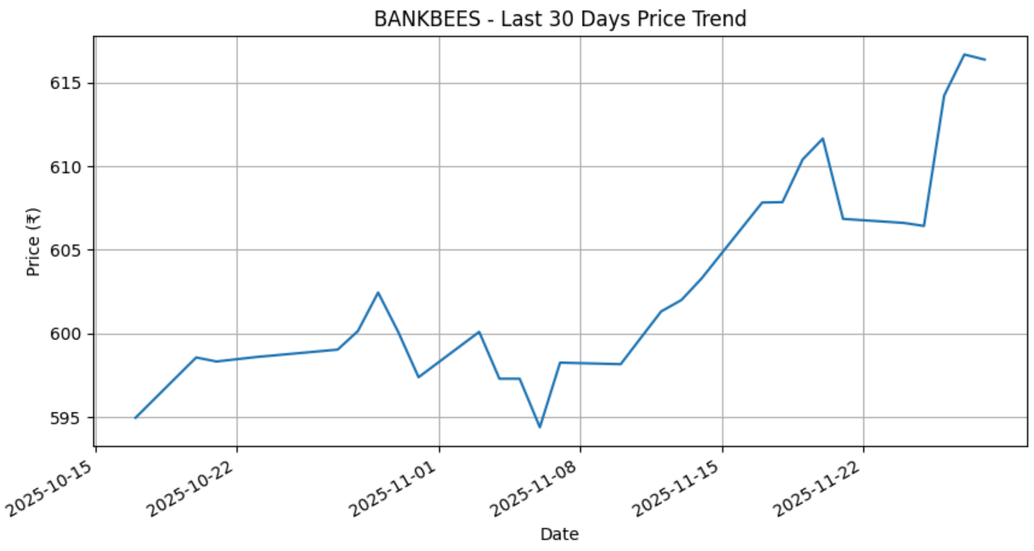
7.4 Graph Output (Description)

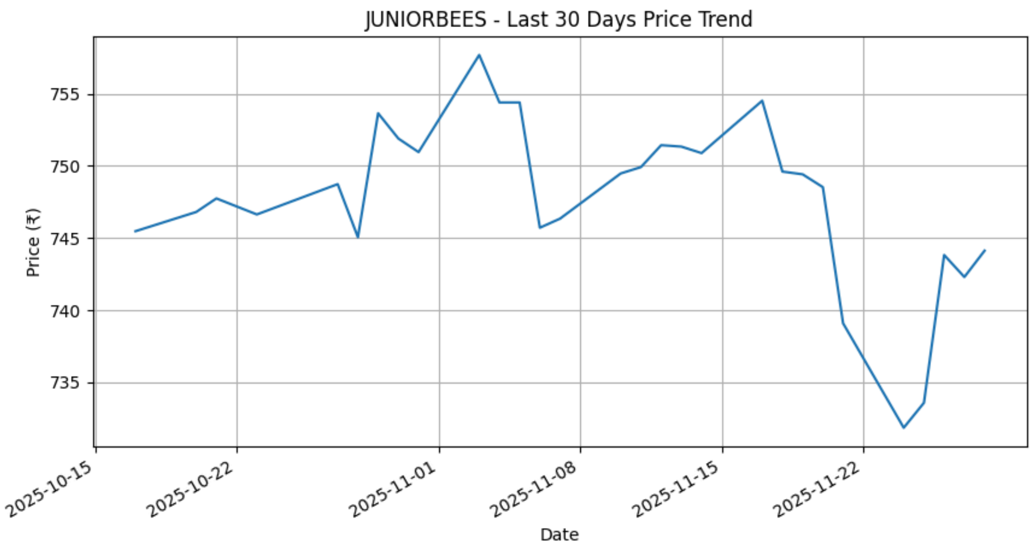
For each ETF, a new **matplotlib window** is opened with:

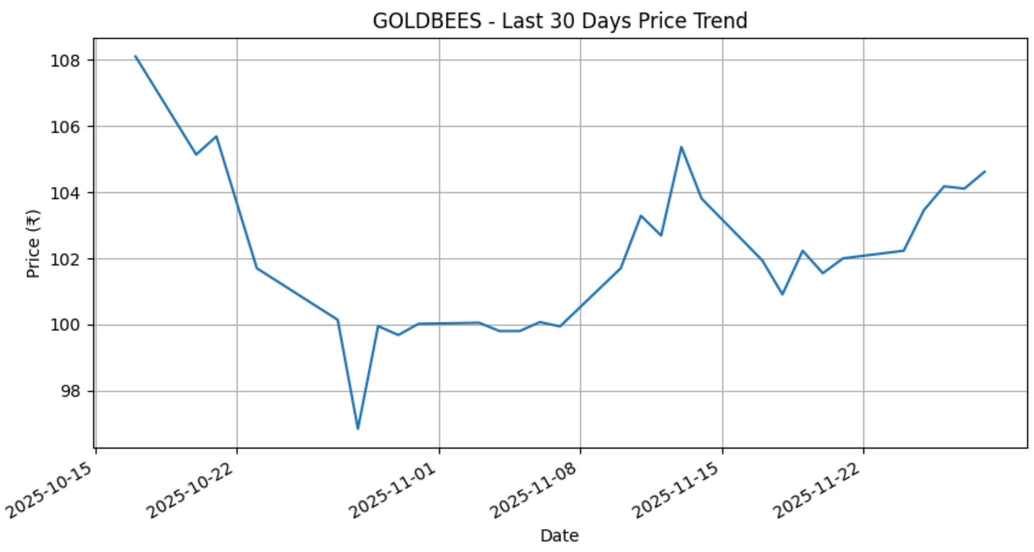
- X-axis: Date
- Y-axis: Close Price

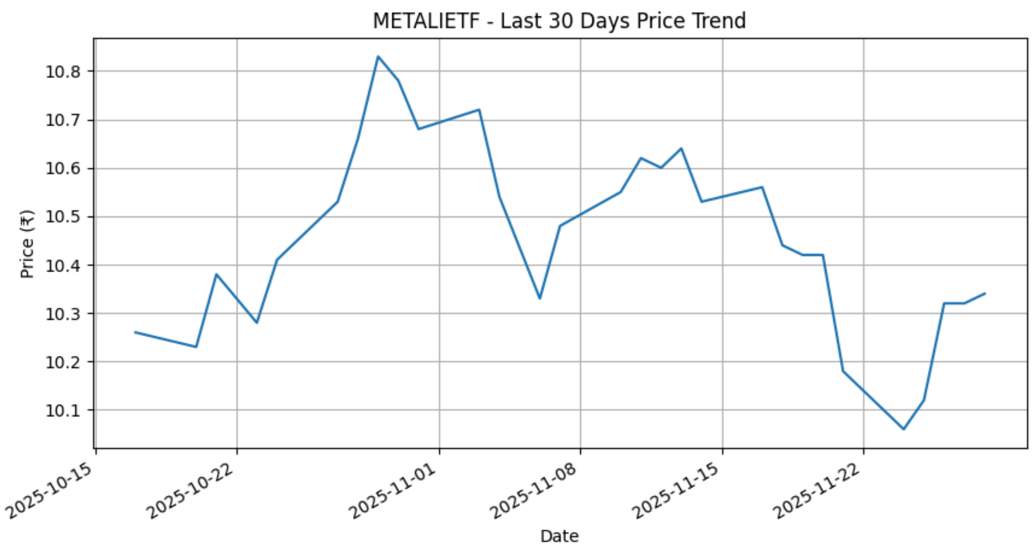
  Below are the Price Trend Graph output:

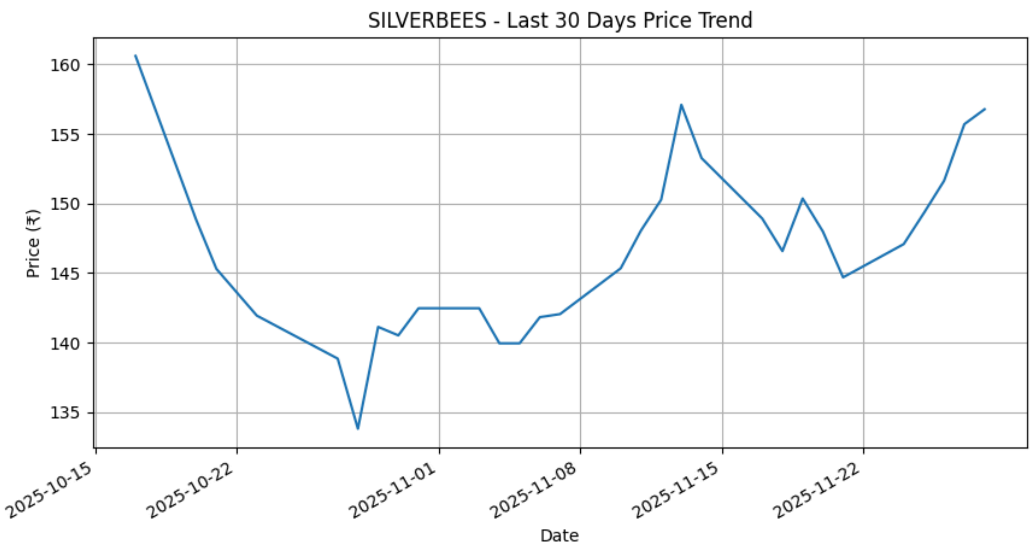

NIFTYBEES - Last 30 Days Price Trend
Price (₹)
Date
296
294
292
290
288
2025-10-15
2025-10-22
2025-11-01
2025-11-08
2025-11-15
2025-11-22

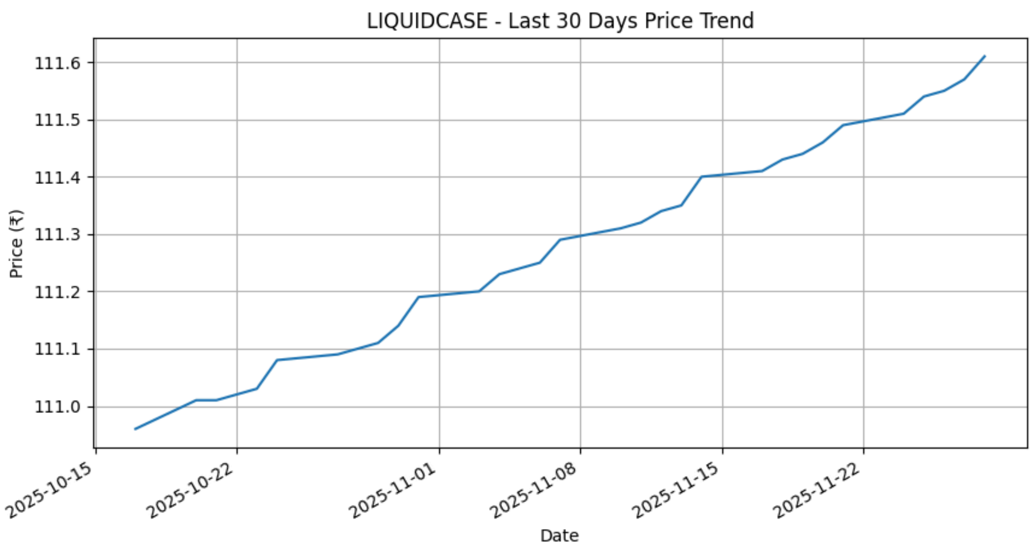

BANKBEES - Last 30 Days Price Trend
Price (₹)
Date
615
610
605
600
595
2025-10-15
2025-10-22
2025-11-01
2025-11-08
2025-11-15
2025-11-22


JUNIORBEES - Last 30 Days Price Trend
Price (₹)
Date
755
750
745
740
735
2025-10-15
2025-10-22
2025-11-01
2025-11-08
2025-11-15
2025-11-22

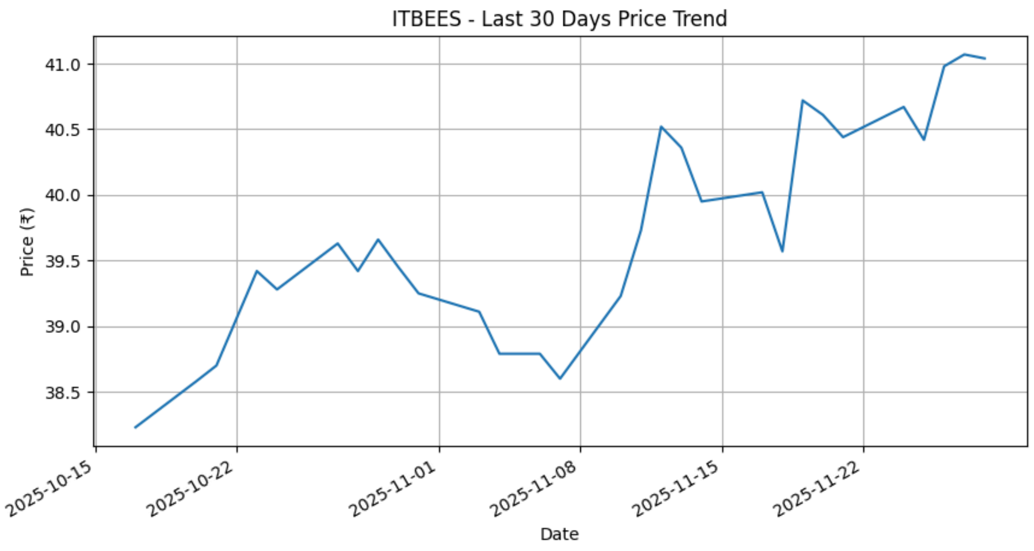

GOLDBEES - Last 30 Days Price Trend
Price (₹)
108
106
104
102
100
98
2025-10-15
2025-10-22
2025-11-01
2025-11-08
2025-11-15
2025-11-22
Date


METALIETF - Last 30 Days Price Trend
Price (₹)
10.8
10.7
10.6
10.5
10.4
10.3
10.2
10.1
2025-10-15
2025-10-22
2025-11-01
2025-11-08
2025-11-15
2025-11-22
Date

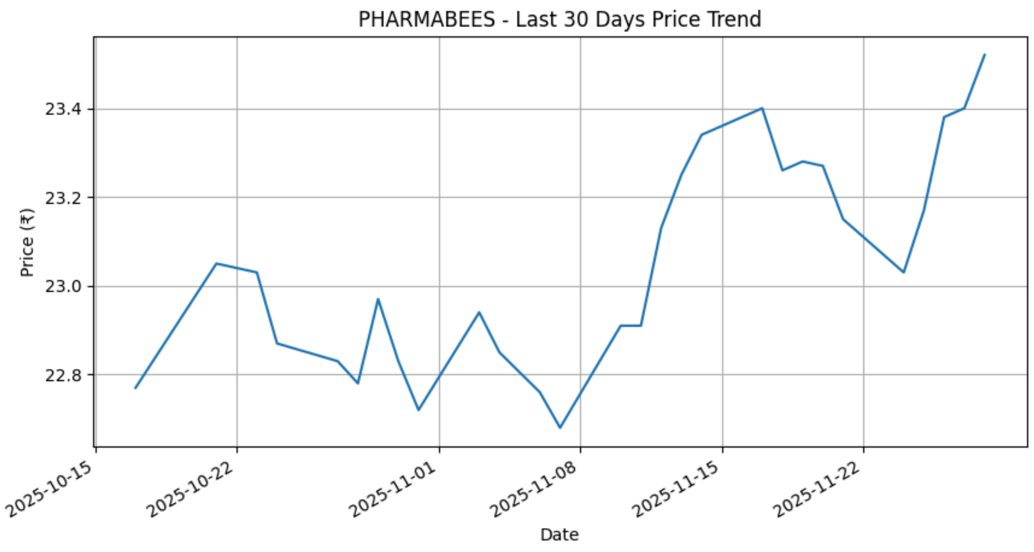

SILVERBEES - Last 30 Days Price Trend
Price (₹)
160
155
150
145
140
135
2025-10-15
2025-10-22
2025-11-01
2025-11-08
2025-11-15
2025-11-22
Date



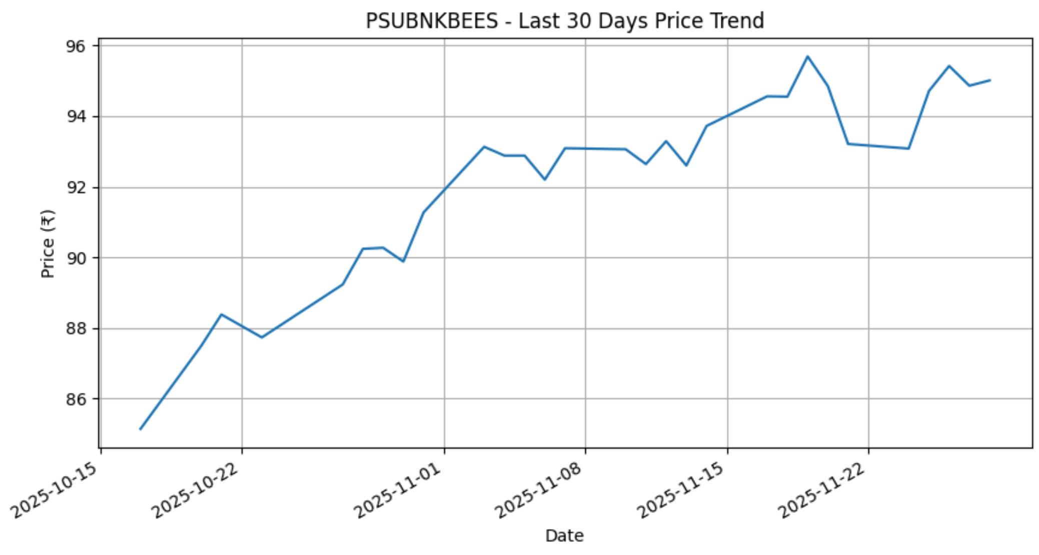
LIQUIDCASE - Last 30 Days Price Trend
Price (₹)
111.0
111.1
111.2
111.3
111.4
111.5
111.6
Date
2025-10-15
2025-10-22
2025-11-01
2025-11-08
2025-11-15
2025-11-22



ITBEES - Last 30 Days Price Trend
Price (₹)
38.5
39.0
39.5
40.0
40.5
41.0
Date
2025-10-15
2025-10-22
2025-11-01
2025-11-08
2025-11-15
2025-11-22



PHARMABEES - Last 30 Days Price Trend
Price (₹)
22.8
23.0
23.2
23.4
Date
2025-10-15
2025-10-22
2025-11-01
2025-11-08
2025-11-15
2025-11-22



- Line gradually moving up or down according to market price movement.

---

8. Conclusion

This mid-level python project, Helps us:

- Quickly check which ETF is **trading at more discount or premium** compared to iNAV.
- Identify better entry opportunities where slippage is less or in favour.
- See **price movement trend** for last N days for all selected ETFs.
- Get the main recommendation table directly in WhatsApp without opening laptop again and again.

It is a simple but practical automation for anyone doing ETF investing.
