Python Code Documentation

get_pe_ratio()

This function, get_pe_ratio, is designed to calculate the Price-to-Earnings (P/E) ratio for a specified stock. The P/E ratio is a financial metric widely used by investors and traders to evaluate the relative value of a company's stock. It represents the relationship between a company's current stock price and its earnings per share (EPS).

Definition of get_pe_ratio

- Function Input: A stock ticker symbol (e.g., "AAPL" for Apple).
- Process: It retrieves data from the Yahoo Finance API via the yfinance library, fetching the current price (current_price) and trailing earnings per share (trailingEps) of the stock.
- Calculation: The P/E ratio is calculated by dividing the current_price by eps.
- Output: The function returns the P/E ratio if available, or None if the required data is missing.

Significance for Traders

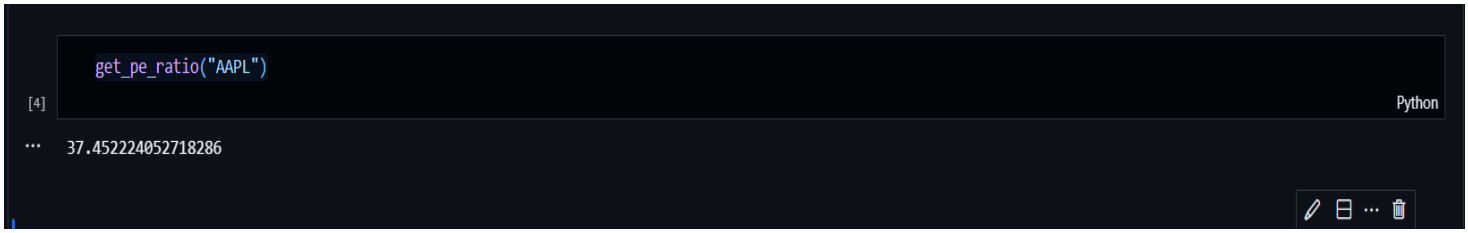
For traders, the P/E ratio is essential for comparing the valuation of companies. A high P/E ratio could suggest that the stock is overvalued or that investors expect high growth rates in the future, while a low P/E might indicate undervaluation or slower growth expectations. By analyzing the P/E ratio, traders can make more informed decisions about whether to buy, hold, or sell a stock.

What the Function Covers

- Fetches Current Price: Retrieves the latest market price of the stock.
- Fetches EPS: Gathers earnings data, which is essential for calculating the P/E ratio.
- Handles Data Availability: Includes error handling in case the data for current price or EPS is unavailable for the given ticker, which is common for certain stocks or newly listed companies.

Overall, this function is a practical tool for traders aiming to quickly assess a stock's valuation relative to its earnings, aiding in fundamental analysis and comparison across stocks.

Use Case



Source Code for the get_pe_ratio() function

```
#Creating the function which takes the ticket as input and returns the price
to earnings ratio

def get_pe_ratio(ticker):
    # Fetch the stock data using yfinance
    stock = yf.Ticker(ticker)

    try:
        # Get the current price
        current_price = stock.info['currentPrice']

        # Get the earnings per share (EPS)
        eps = stock.info['trailingEps']

        # Calculate the P/E ratio
        pe_ratio = current_price / eps

        return pe_ratio
    except KeyError:
        print(f"Unable to calculate P/E ratio for {ticker}. Required data not
available.")
        return None
```

get_pb_ratio(ticker)

Significance of the Financial Metric: Price-to-Book (P/B) Ratio

The Price-to-Book (P/B) ratio is a financial metric that compares a company's market value to its book value. It is calculated as:

P/B=Market Price per ShareBook Value per ShareP/B = \frac{\text{Market Price per Share}}{\text{Book Value per Share}}P/B=Book Value per ShareMarket Price per Share

- Market Price per Share: The current trading price of a single share of the company.
- Book Value per Share: The value of the company's net assets (total assets minus liabilities) divided by the total number of outstanding shares.

Why is the P/B Ratio Important?

1. Valuation Indicator:
    - A low P/B ratio (< 1) may indicate that the stock is undervalued relative to its book value.
    - A high P/B ratio may suggest overvaluation or that investors expect high growth.
2. Industry-Specific Insight: The P/B ratio is more relevant for asset-heavy industries like banking, real estate, or manufacturing.
3. Risk Assessment: It provides insights into the company's financial health and investor sentiment.

Source Code

```
def get_pb_ratio(ticker):
    # Fetch the stock data using yfinance
    stock = yf.Ticker(ticker)

    try:
        # Get the current price
        current_price = stock.info['currentPrice']

        # Get the book value per share
        book_value_per_share = stock.info['bookValue']

        # Calculate the P/B ratio
        pb_ratio = current_price / book_value_per_share

        return pb_ratio
    except KeyError:
```

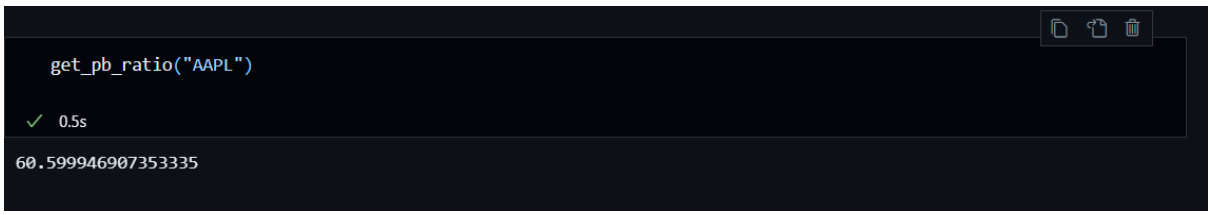
```
        print(f"Unable to calculate P/B ratio for {ticker}. Required data not
available.")
        return None

# Testing the created function
#print(get_pb_ratio("AAPL"))
```

Explanation of the Function

The function get_pb_ratio retrieves the P/B ratio for a specified stock ticker using the yfinance library. Here's a breakdown:

1. Input:
    - ticker: The stock's ticker symbol (e.g., "AAPL" for Apple Inc.).
2. Fetching Stock Data:
    - The yfinance library's Ticker object fetches detailed stock information for the specified ticker.
3. Data Retrieval:
    - The currentPrice and bookValue are extracted from the info dictionary provided by yfinance.
4. P/B Ratio Calculation:
    - If the required data (currentPrice and bookValue) is available, the P/B ratio is calculated using: P/B Ratio=currentPricebookValue\text{P/B Ratio} = \frac{\text{currentPrice}}{\text{bookValue}}P/B Ratio=bookValuecurrentPrice
5. Error Handling:
    - If any of the required fields (currentPrice or bookValue) are missing, a KeyError exception is handled gracefully. A message is printed indicating that the data is unavailable, and the function returns None.
6. Output:
    - Returns the calculated P/B ratio if successful, or None if the data is not available.

Use Case



get_ps_ratio(ticker)

Significance of the Financial Metric: Price-to-Sales (P/S) Ratio

The Price-to-Sales (P/S) ratio is a financial metric that measures the value investors place on a company's sales relative to its market value. It is calculated as:

P/S=Market Price per ShareRevenue per ShareP/S = \frac{\text{Market Price per Share}}{\text{Revenue per Share}}P/S=Revenue per ShareMarket Price per Share

- Market Price per Share: The current trading price of a single share of the company.
- Revenue per Share: The total revenue of the company divided by the number of outstanding shares.

Why is the P/S Ratio Important?

1. **Valuation Benchmark:**
    - A low P/S ratio may indicate that a company is undervalued compared to its sales.
    - A high P/S ratio may suggest that investors expect significant future growth.
2. **Comparative Analysis:** Useful for comparing companies in the same industry, especially for firms with no profits (e.g., startups or growth companies).
3. **Industry Neutrality:** While more relevant for revenue-driven industries, the P/S ratio can be applied across a broad range of sectors.

Explanation of the Function

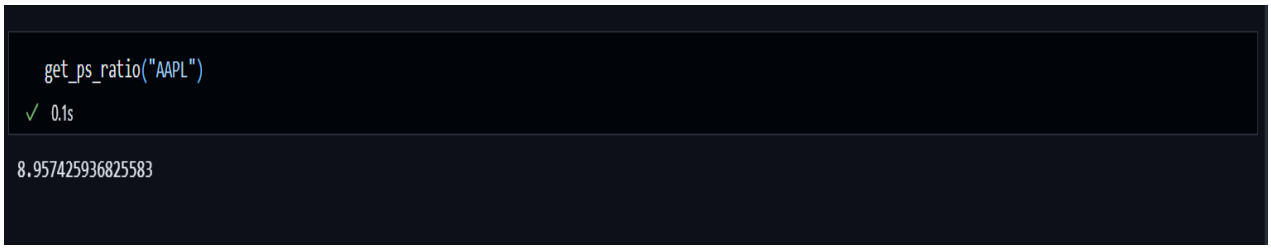
The function get_ps_ratio retrieves the P/S ratio for a specified stock ticker using the yfinance library. Here's how it works:

1. **Input:**
    - ticker: The stock's ticker symbol (e.g., "AAPL" for Apple Inc.).
2. **Fetching Stock Data:**
    - The yfinance library's Ticker object fetches comprehensive stock information for the specified ticker.
3. **Data Retrieval:**
    - The currentPrice and revenuePerShare fields are extracted from the info dictionary provided by yfinance.
4. **P/S Ratio Calculation:**
    - If the required data (currentPrice and revenuePerShare) is available, the P/S ratio is calculated using: P/S Ratio=currentPricerevenuePerShare\text{P/S Ratio} =

\frac{\text{currentPrice}}{\text{revenuePerShare}}P/S Ratio=revenuePerSharecurrent Price

5. **Error Handling:**
    - If either currentPrice or revenuePerShare is unavailable, the function handles the KeyError gracefully. It prints a message and returns None.
6. **Output:**
    - Returns the calculated P/S ratio if successful, or None if the data is unavailable.

Use Case



Source Code

```
def get_ps_ratio(ticker):
    # Fetch the stock data using yfinance
    stock = yf.Ticker(ticker)

    try:
        # Get the current price
        current_price = stock.info['currentPrice']

        # Get the revenue per share
        revenue_per_share = stock.info['revenuePerShare']

        # Calculate the P/S ratio
        ps_ratio = current_price / revenue_per_share

        return ps_ratio
    except KeyError:
        print(f"Unable to calculate P/S ratio for {ticker}. Required data not
available.")
        return None

# Testing the created function
#print(get_ps_ratio("AAPL"))
```

get_ps_ratio(ticker)

Significance of the Financial Metric: Enterprise Value (EV)

Enterprise Value (EV) is a key financial metric that represents the total value of a company. It is a comprehensive measure that includes equity value, debt, and cash, providing a clear picture of the firm's financial health and valuation. The formula is:

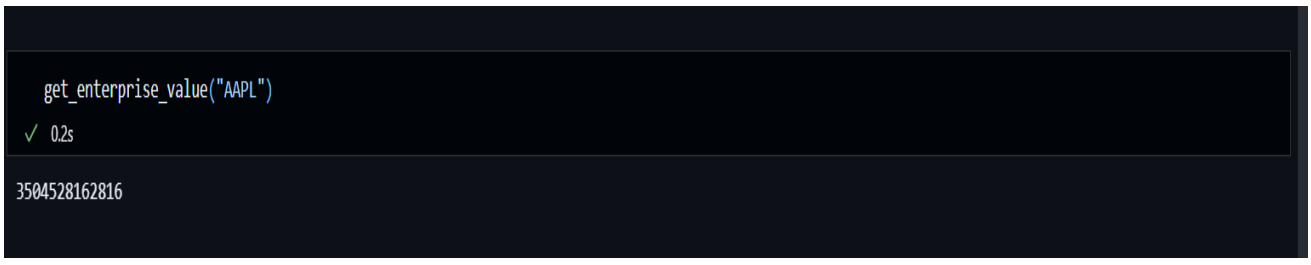
EV=Market Capitalization+Total Debt−Cash and Cash EquivalentsEV = \text{Market Capitalization} + \text{Total Debt} - \text{Cash and Cash Equivalents}EV=Market Capitalization+Total Debt−Cash and Cash Equivalents

However, some financial data platforms like yfinance directly provide the pre-calculated **Enterprise Value** to streamline analysis.

Why is EV Important?

1. **Complete Valuation**:
    - Unlike market capitalization, EV incorporates debt and cash to give a fuller view of a company's worth.
2. **Capital Structure-Neutral**:
    - Useful for comparing companies with different financing structures (e.g., debt-heavy vs. equity-heavy firms).
3. **M&A Valuation**:
    - Frequently used in mergers and acquisitions to assess the true cost of acquiring a business.

Use Case



Source Code

```
def get_enterprise_value(ticker):
    # Fetch the stock data using yfinance
    stock = yf.Ticker(ticker)

    try:
        # Get the enterprise value
        enterprise_value = stock.info['enterpriseValue']

        return enterprise_value
    except KeyError:
        print(f"Unable to retrieve Enterprise Value for {ticker}. Required
data not available.")
        return None

# Testing the created function
#print(get_enterprise_value("AAPL"))
```

Explanation of the Function

1. **Input**:
    - ticker: The stock's ticker symbol (e.g., "AAPL" for Apple Inc.).
2. **Fetching Stock Data**:
    - The yfinance library's Ticker object retrieves comprehensive stock information for the specified ticker.
3. **Data Retrieval**:
    - The function accesses the enterpriseValue key from the info dictionary provided by yfinance. This value represents the pre-calculated Enterprise Value.
4. **Output**:
    - Returns the **Enterprise Value** if the data is successfully retrieved.
5. **Error Handling**:
    - If the enterpriseValue key is unavailable in the stock data, a KeyError is caught. The function prints an error message and returns None.

get_ev_ebitda(ticker)

The **Enterprise Value to EBITDA (EV/EBITDA)** ratio is a widely used valuation metric that compares a company's total value (EV) to its Earnings Before Interest, Taxes, Depreciation, and Amortization (EBITDA). It provides a measure of how much an investor is willing to pay for a company's operational performance. The formula is:

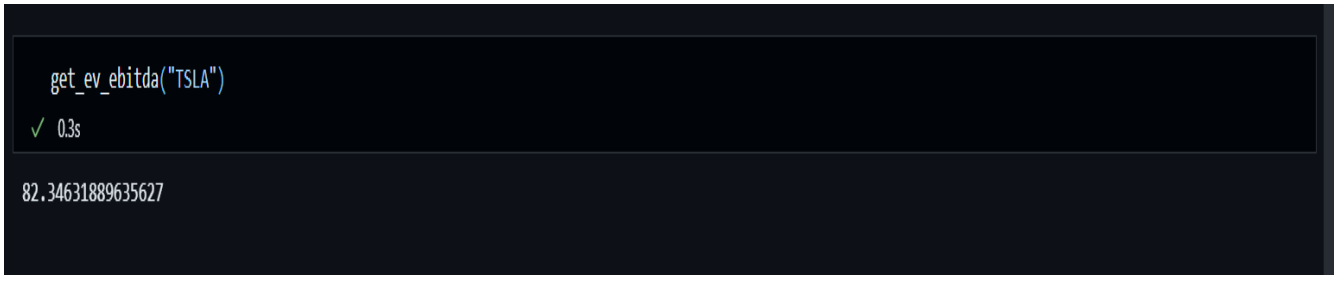
EV/EBITDA=Enterprise Value (EV)EBITDA\text{EV/EBITDA} = \frac{\text{Enterprise Value (EV)}}{\text{EBITDA}}EV/EBITDA=EBITDAEnterprise Value (EV)

- **Enterprise Value (EV)**: The total valuation of a company, including equity, debt, and cash adjustments.
- **EBITDA**: A proxy for cash flow from operations, excluding non-operating expenses.

Why is EV/EBITDA Important?

1. **Valuation Benchmark**:
    - A low EV/EBITDA ratio may indicate that a company is undervalued.
    - A high EV/EBITDA ratio suggests higher valuation expectations, possibly due to strong growth potential.
2. **Cross-Industry Comparison**:
    - Suitable for comparing companies across industries with different capital structures.
3. **Operational Focus**:
    - Unlike other ratios, it focuses on operational performance without being affected by financing or accounting policies.

Use Case



Source Code

```
def get_ev_ebitda(ticker):
    # Fetch the stock data using yfinance
    stock = yf.Ticker(ticker)

    try:
        # Get the Enterprise Value
        enterprise_value = stock.info['enterpriseValue']

        # Get the EBITDA
        ebitda = stock.info['ebitda']

        # Calculate the EV/EBITDA ratio
        ev_ebitda_ratio = enterprise_value / ebitda

        return ev_ebitda_ratio
    except KeyError:
        print(f"Unable to calculate EV/EBITDA ratio for {ticker}. Required
data not available.")
        return None
```