```python
"""
Fetching and formatting stock market data, including trending tickers and key statistics.

This module provides functions to:
1. Retrieve a list of trending stock tickers from Yahoo Finance.
2. Format financial metrics such as Market Cap and Enterprise Value.
3. Fetch and format key financial statistics for a list of tickers.
4. Generate HTML components for displaying ticker information.

Variables:
- TOOLTIP_TEXT (dict): A dictionary containing explanatory text for various financial
metrics.

Functions:
- get_trending_tickers()
- format_market_cap(value)
- format_enterprise_value(value)
- get_key_statistics(tickers)
- generate_ticker_rectangles()
"""

TOOLTIP_TEXT = {
    "Market Cap": "The total market value of a company's outstanding shares. Indicates the
size of the company and is used to compare companies within the same industry.",
    "Trailing P/E": "Price-to-earnings ratio based on the last 12 months of actual earnings.
Helps investors understand how much they are paying for a company's earnings. A high P/E
might indicate high future growth expectations, while a low P/E might indicate the
opposite.",
    "PEG Ratio": "Price/earnings-to-growth ratio, which factors in expected earnings growth.
Helps determine if a stock is over or undervalued considering its earnings growth",
    "Price/Sales": "The ratio of a company's stock price to its revenues. Useful for
evaluating companies that are not yet profitable. It shows how much investors are willing to
pay per dollar of sales.",
    "Enterprise Value": "The total value of a company, including debt and excluding cash.",
    "EV/Revenue": "The ratio of enterprise value to revenue.  Indicates how much investors
are willing to pay for each dollar of revenue, providing insight into a company's valuation
relative to its sales."
}

def get_trending_tickers():
    """
    Fetches trending stock tickers from Yahoo Finance.

    Returns:
        list: A list of dictionaries containing ticker symbols of trending stocks.
    """
    url = "https://finance.yahoo.com/trending-tickers/"
    try:
        response = requests.get(url)
        response.raise_for_status()  # Raise an exception for bad response status codes

        soup = BeautifulSoup(response.content, "html.parser")
        table = soup.find('div', {'id': 'list-res-table'})  # Update class name here

        if table is None:
            raise ValueError("Unable to find table with class 'W100'")

        rows = table.find_all('tr')

        trending_tickers_data = []
        for row in rows[1:]:  # skipping header row
            columns = row.find_all("td")
            ticker = columns[0].text.strip()

            trending_tickers_data.append({
                "Ticker": ticker
            })
        return trending_tickers_data

    except requests.RequestException as e:
        print(f"Error fetching data from {url}: {e}")
        return []
    except Exception as e:
        print(f"Error: {e}")
        return []

def format_market_cap(value):
    """
    Formats the market capitalization value into a readable string.

    Args:
        value (str): The market cap value as a string.

    Returns:
        str: Formatted market cap with appropriate suffix (T, B, M) or "N/A" if not
available.
    """
    if value == "N/A":
        return value

    value = float(value)
    if value >= 1e12:  # Trillions
        return f'{value / 1e12:.3f}T'
    elif value >= 1e11:  # Hundreds of Billions
        return f'{value / 1e9:.3f}B'
    elif value >= 1e10:  # Tens of Billions
        return f'{value / 1e9:.3f}B'
    elif value >= 1e9:  # Billions
        return f'{value / 1e9:.3f}B'
    elif value >= 1e6:  # Millions
        return f'{value / 1e6:.3f}M'
    else:
        return str(value)

def format_enterprise_value(value):
    """
    Formats the enterprise value into a readable string.

    Args:
        value (str): The enterprise value as a string.

    Returns:
        str: Formatted enterprise value with appropriate suffix (T, B, M) or "N/A" if not
available.
    """
    if value == "N/A":
        return value

    value = float(value)
    if value >= 1e12:  # Trillions
        return f'{value / 1e12:.2f}T'
    elif value >= 1e11:  # Hundreds of Billions
        return f'{value / 1e9:.2f}B'
    elif value >= 1e10:  # Tens of Billions
        return f'{value / 1e9:.2f}B'
    elif value >= 1e9:  # Billions
        return f'{value / 1e9:.2f}B'
    elif value >= 1e6:  # Millions
        return f'{value / 1e6:.2f}M'
    else:
        return str(value)

def get_key_statistics(tickers):
    """
    Retrieves key financial statistics for a list of tickers.

    Args:
        tickers (list of str): List of stock tickers.

    Returns:
        dict: A dictionary where each key is a ticker symbol and each value is another
dictionary containing key statistics.
    """
    statistics = {}
    for ticker in tickers:
        ticker_data = yf.Ticker(ticker)
        stats = ticker_data.info
        key_stats = {
            "Market Cap": format_market_cap(stats.get("marketCap", "N/A")),
            "Trailing P/E": round(stats.get("trailingPE", "N/A"), 2) if
stats.get("trailingPE", "N/A") != "N/A" else "N/A",
            "PEG Ratio": round(stats.get("pegRatio", "N/A"), 2) if stats.get("pegRatio",
"N/A") != "N/A" else "N/A",
            "Price/Sales": round(stats.get("priceToSalesTrailing12Months", "N/A"), 2) if
stats.get("priceToSalesTrailing12Months", "N/A") != "N/A" else "N/A",
            "Enterprise Value": format_enterprise_value(stats.get("enterpriseValue",
"N/A")),
            "EV/Revenue": round(stats.get("enterpriseToRevenue", "N/A"), 2) if
stats.get("enterpriseToRevenue", "N/A") != "N/A" else "N/A"
        }
        statistics[ticker] = key_stats
    return statistics

def generate_ticker_rectangles():
    """
    Generates a list of styled HTML div elements representing the trending tickers.

    Returns:
        list: A list of HTML div elements, each containing a ticker symbol.
    """
    tickers = get_trending_tickers()[:24]  # Limit to the first 24 tickers
    max_ticker_length = max(len(ticker['Ticker']) for ticker in tickers)

    return [
        html.Div(
            ticker['Ticker'],
            style={
                'padding': '20px',
                'margin': '10px',
                'backgroundColor': 'rgba(47, 79, 79, 0.6)',  # Slate theme background color
with 60% opacity
                'color': '#00ffff',
                'borderRadius': '10px',
                'boxShadow': '2px 2px 5px rgba(0, 0, 0, 0.3)',
                'textAlign': 'center',
                'width': f'{max_ticker_length * 15}px',  # Adjust width based on ticker
length
                'fontFamily': 'Lato, monospace',
                'font-weight': 'bold',
                'display': 'flex',  # Use Flexbox
                'justifyContent': 'center',  # Center horizontally
                'alignItems': 'center'  # Center vertically
            }
        ) for ticker in tickers
    ]

# Load the SLATE theme
from dash_bootstrap_templates import load_figure_template
load_figure_template("SLATE")
```