```python
"""
This module contains functions and imports for financial data analysis and visualization.

Key components include:

- **Data Download and Processing**: Functions to download historical stock data, categorize tickers based on data availability, and handle missing values.
- **Data Handling Libraries**: Utilizes libraries such as `yfinance`, `pandas`, and `numpy` for data manipulation.
- **Statistical Analysis**: Implements statistical analysis and optimization using libraries like `scipy` and `statsmodels`.
- **Visualization**: Integrates `matplotlib` and `plotly` for plotting and visualizing financial data.
- **Web and API Interactions**: Includes modules for web scraping (`requests`, `BeautifulSoup`) and web application frameworks (`dash`, `dash_bootstrap_components`).
- **Forecasting**: Utilizes `pmdarima` for automated ARIMA model fitting.

Functions:
- `download_data_fillna(tickers, start_date, end_date)`: Downloads historical stock data, categorizes tickers based on data availability, and handles missing values by forward-filling.
"""

import yfinance as yf
import pandas as pd
import numpy as np
from yahoo_fin import stock_info as si
from datetime import datetime, timedelta
from scipy.optimize import minimize
from itertools import product
import dash_bootstrap_components as dbc
from dash import html, dcc, dash_table
from pmdarima import auto_arima
from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt
import plotly.graph_objs as go
import dash
from dash.dependencies import Input, Output, State, ALL
import requests
from bs4 import BeautifulSoup
from dash_bootstrap_templates import load_figure_template
import json
from io import StringIO
from dash.exceptions import PreventUpdate


combined_data = None

def download_data_fillna(tickers, start_date, end_date):
    """
    Download historical stock data for a list of tickers, categorize tickers based on the data availability, and handle missing values.

    This function downloads closing price data for each ticker between the specified start and end dates using the `yfinance` library. It categorizes tickers into valid, young, or invalid based on the availability of data and the age of the data. Missing values in the data are forward-filled to ensure continuity.

    Args:
        tickers (list of str): A list of stock tickers to download data for.
        start_date (str): The start date for the data download in 'YYYY-MM-DD' format.
        end_date (str): The end date for the data download in 'YYYY-MM-DD' format.

    Returns:
        tuple:
            - pd.DataFrame: A DataFrame containing the closing price data for valid tickers with forward-filled missing values.
            - list of str: A list of tickers categorized as young due to insufficient
historical data.
            - list of str: A list of tickers categorized as invalid due to errors or lack of
data.
    """
    valid_tickers = []
    young_tickers = []
    invalid_tickers = []

    for ticker in tickers:
        try:
            data = yf.download(tickers=[ticker], start=start_date, end=end_date)['Close']
            print(f"Downloaded data for {ticker}: {data}")

            if data.empty:
                print(f"{ticker} has no data, categorizing as invalid.")
                invalid_tickers.append(ticker)
                continue

            if data.index[0] > pd.to_datetime("2023-01-03"):
                print(f"{ticker} is young, categorizing as young.")
                young_tickers.append(ticker)
            else:
                data = pd.DataFrame(data.rename(ticker))
                valid_tickers.append(data)
                print(f"{ticker} is valid and added to the list of valid tickers.")

        except Exception as e:
            print(f"Error downloading data for {ticker}: {e}")
            invalid_tickers.append(ticker)

    if valid_tickers:
        all_data = pd.concat(valid_tickers, axis=1)
        all_data = all_data.asfreq('B')  # Ensure data is business day frequency

        data = all_data.ffill()

        return data, young_tickers, invalid_tickers
    else:
        return pd.DataFrame(), young_tickers, invalid_tickers
```