Operating instructions Python

Usage Guide:

To use this program, follow these steps:

1. First, ensure you have the necessary Python packages installed: **matplotlib**, **yfinance**, **pandas**, **numpy**, **scipy**, **fpdf**, **seaborn**, and **reportlab**. Install using pip (**pip install package-name**).
2. Import the provided Python class into your script: **from your_script import Portfolio**.
1. Create a portfolio by passing in a dictionary where the keys are the asset tickers and the values are the quantities: **assets = {"AAPL": 10, "MSFT": 5}; ptf = Portfolio(assets, start="YYYY-MM-DD", end="YYYY-MM-DD")**. Prepare an Excel file with the portfolio information. The file should have two columns: "Tickers" and "Qté". Each row represents one asset in the portfolio, with the ticker symbol and quantity.
2. Use the "Browse" button to select the Excel file.
3. Use the **ptf._download_prices()** method to download prices for the assets within the specified date range.
4. Calculate returns, covariance matrix, portfolio returns, portfolio volatility with respective methods: **ptf.calculate_assets_returns()**, **ptf.calculate_cov_matrix()**, **ptf.calculate_ptf_returns()**, **ptf.calculate_portfolio_volatility()**.
5. To measure the portfolio's performance, use **ptf.performance_ytd()**, **ptf.performance_1_week()**, **ptf.performance_1_month()**, **ptf.performance_1_year()**.
6. Use **ptf.bench_returns()** to get returns of a benchmark.
7. To analyze risks, use **ptf.historical_var()**, **ptf.parametric_var()**, **ptf.monte_carlo_var()**.
8. Visualize your portfolio information using **ptf.plot_portfolio_info2()**.
3. Finally, by clicking on the "Analyze Portfolio" button to execute the analysis.you can export your analysis to a PDF using **ptf.export_to_pdf()**. The results will be displayed on the console, and a PDF report will be saved in the current directory.

Note: Modify the start and end dates to suit your needs. "YYYY-MM-DD" is a placeholder for dates in the format "2022-12-31". Remember to replace **'your_script'** with the actual name of your python script.

Our code provides a robust tool for portfolio analysis and visual representation of portfolio characteristics. It's written in Python and relies on several libraries to execute its operations. Here's an overview of its functions:

1. **Class Definition - Portfolio:** The core of the program, this class includes several methods for portfolio analysis. You initiate a portfolio by providing a dictionary of assets (with tickers as keys and quantities as values) and optional start and end dates for the analysis.
2. **Data Download:** The class uses the **yfinance** library to download historical prices for the assets in the portfolio. The prices are stored in a DataFrame, which is then used in various calculations.
3. **Portfolio Statistics:** Several methods allow for the calculation of important portfolio statistics, including returns, volatility, Value-at-Risk (VaR), and benchmark comparison.
    - **Asset returns** are calculated based on price changes, and portfolio returns are calculated as the weighted sum of asset returns.
    - **Volatility** is computed as the standard deviation of portfolio returns.
    - **Value-at-Risk (VaR)** is calculated in three different ways: Historical VaR, Parametric VaR, and Monte Carlo VaR.
    - **Benchmark comparison** calculates the alpha and beta of the portfolio relative to a benchmark index.
4. **Visualization:** The **plot_portfolio_info2** method plots several key portfolio characteristics including portfolio value, returns, and performance relative to a benchmark. It also includes a summary of key metrics.
5. **Exporting to PDF:** The **export_to_pdf** method creates a PDF report of the portfolio analysis, which includes the plot generated by **plot_portfolio_info2** and a brief introduction.
6. **Graphical User Interface (GUI):** The **PortfolioApp** class provides a simple GUI for users to select their portfolio Excel file and execute the analysis.