Week2 Assignment

Develop a Python Emailer for Market Data Monitor

Objective

Build up a Python based emailer, which can send email with attachments, plots, and content in the email body. Leveraging the emailer to automate the daily process of retrieving stock price data and deliver an email to the team containing:

1. An Excel attachment with stock price data from 2023.
2. A summary table for Year-to-Date (YTD), Month-to-Date (MTD), and Day-to-Date (DTD) returns.
3. Graphs showing price trends.

Procedures

Step 1: Set Up Gmail (please use your personal email rather than Stratifi Email) for Sending Emails

- Enable Two-Step Verification: Go to your Google account settings and enable 2-Step Verification. This is needed for secure email access.
- Create an App Password: After enabling 2-Step Verification, create an App Password (this will be used in your Python script to send emails).

Step 2: Implement the Email Sending Function

- You'll use Python's `smtplib` libraries to send emails.
- Create a function called `send_email()` to handle sending emails.
- Sends an email to the specified recipients; Supports HTML-formatted content for enhanced styling; Allows attaching files and plots as part of the email.
- Function signature:
  `def send_email(recipients, subject, content, attachments=None, plots=None)`
- Parameters:
  1. recipients (string)
     - Description: Email address(es) of the recipient(s). Multiple addresses should be separated by semicolons (;).
     - Example: “mike_yu@stratifisolutions.com; jacky_z@stratifisolutions.com”
  2. content (string)
     - Description: HTML-formatted body of the email for styled content or rich text.
     - Example: "<html><body><h1>Daily Report</h1><p>Here is your report...</p></body></html>"
  3. attachments (list, optional)
     - Description: A list of file paths for files to attach to the email. Supports documents, spreadsheets, PDFs, etc.
     - Default: None
     - Example: ["/path/to/report.pdf", "/path/to/data.csv"]

4. plots (list, optional)
    - Description: A list of file paths for images or plots to include as additional attachments.
    - Default: None
    - Example: ["/path/to/plot1.png", "/path/to/plot2.jpg"]

Step 3: Create an HTML Template for the Email Body

- Create an HTML template that structures the email content neatly, making it visually appealing and easy to read.
- Content Structure:
  a. Display the YTD, MTD, and DTD returns for SPX, VIX, and AAPL in a formatted table.
  b. Insert the time-series line plots directly into the email for a quick visual overview.

Step 3: Retrieve Data and Generate Content

1. Download Price Data: Use the yfinance API to download daily price data for SPX, VIX, and AAPL starting from January 2, 2023
2. Calculate Returns: Assume today is December 27, 2024, and calculate:
    - DTD Return (Day-to-Date): Percent change from the previous trading day.
    - MTD Return (Month-to-Date): Percent change from the first trading day of December 2024.
    - YTD Return (Year-to-Date): Percent change from the first trading day of 2024.
3. Generate Graphs: Create line graphs showing the price trends in 2024 for each ticker. Save these as jpg files.
4. Create Excel Report: Save the daily price data for each ticker into an Excel file (xlsx) with three sheets (one per ticker).

Step 4: Create an HTML Template for the Email Body

- Create an HTML template that structures the email content neatly, making it visually appealing and easy to read.
- Content Structure: Display the YTD, MTD, and DTD returns for SPX, VIX, and AAPL in a formatted table. Insert the time-series line plots directly into the email body for a quick visual overview.

Step 5: Assemble and Send the Email

- Call the `send_email` function with the generated HTML body, attaching the Excel file and graphs, and send the email to the recipient.
- Schedule this process to run daily for automated reporting. You can use **Task Scheduler** (Windows) or **Scheduler** (Mac) to trigger your Python script at your scheduled time.