The Brief

Automation Objectives

Form a team of not more than 5 per team and write a business automation program:

1. Make an API call to https://www.alphavantage.co to extract the real time exchange rate (Forex) in JSON output using the following API parameters:
    - function=CURRENCY_EXCHANGE_RATE
    - from_symbol=USD
    - to_symbol=SGD

2. Extract and summarise data from the finance dashboard in the final round of business stimulation game in MAB module. Refer to PDF : Instructions to download CSV files to download the files from the game.

3. The automation will perform tasks from the following csv files:

    a. Profit & Loss csv : The program will compute the **difference in the net profit** between each day. If the net profit is not consecutively higher each day, the program will highlight the day where net profit is lower than the previous day and the value difference.
    b. Cash-On-Hand csv : The program will compute the **difference in Cash-on-Hand** between each day. If Cash-on-Hand is not consecutively higher each day, the program will highlight the day where Cash-on-Hand is lower than the previous day and the value difference.
    c. Overheads csv : The program will find the **highest** overhead category and its value.
    d. The program will convert the computed amount from task **a, b and c** using the real time exchange rate from the API call.
    e. Write the computed amount from task **a to d** to a text file and name it : summary_report.txt.

Figure 1.0 included two scenarios to illustrate the automation objectives and the expected output in summary_report.txt.

Exception Handling

The program is expected to include exception handling in the areas your team deem necessary. This will help to maintain the normal, desired flow of the program even when unexpected events occur. If exceptions are not handled, programs may crash or requests may fail.

The Brief

Files and Project Directory

You should organise your program and csv files into the following folder structure:

Dedicate each python file to achieve a specific task. For example, api.py should only contain code that finds the real time exchange rate, while overheads.py should only contain codes that find the highest overhead category.

Organizing code this way will make the overall program more manageable, easier to maintain and debug errors.

Note that file path specified in each python file should be directed to **current working directory** instead of your computer's directory. When your program is being evaluated, it will be based on current working directory.

Required Coding Skills

To complete the assignment successfully, you need to use only the programming topics learn from PFB, unless given the permission to do so.

The use of external modules not taught will severely affect the grade. External module refers to additional module installed with pip install command.

However, you may use any **built-in** functions or/and modules.

Project Standard Criteria

Your code solution will be evaluated based on the following criteria:

1. Program Correctness
2. Code Readability
3. Code Elegance/ Efficiency
4. Code Documentation
5. Assignment Specification

Project Bonus Marks

Bonus marks will be awarded based on the group's ability to:

1. Collaborate on coding with GitHub
2. Modularization of program

The Brief

How to Collaborate?

1. As this is a group project, you are required to collaborate with code with each other and each member is expected to contribute to the project. To collaborate coding better, you can make of use **GitHub**, a leading collaboration platform used by major tech companies and programmers worldwide.

2. A set of instructional slides on how to collaborate on GitHub using Visual Studio Code are available. Refer to PDF : Collaborate with GitHub

3. Each member should be assigned to work on a specific part of the program. For example, a team member can work on the api.py, while another member can work on the profit_loss.py.

What is a modular program?

1. Modularization is the technique of splitting a large programming task into smaller, separate, and manageable subtasks.
2. To achieve modularization, you can further organized the code in each python file as a function.
3. A main python file (main.py) will import these functions, to coordinate and execute the functions.
4. In this way the overall program becomes even more manageable, easier to maintain and debug errors.
5. Refer to Figure 2.0 for an example of modularizing a complex program.

Figure 1.0 Automation Objectives

SCENARIO 1:

1. From the Overheads.csv, Salary Expense is the highest overheads.
2. From the "Cash on Hand.csv" and "Profit & Loss.csv", the values are consecutively higher than the previous day.

Overheads.csv

Category	Overheads
Salary Expense	28.77
Interest Expense	0.23
Rental Expense	20.64
Penalty Expense	12.88
Depreciation Expense	20.83
Human Resource Expense	16.66

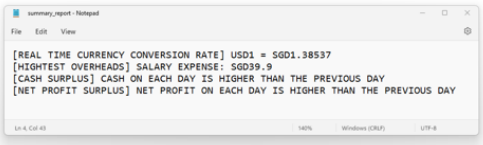
Cash on Hand.csv

Cash On Hand	
Day	Cash On Hand
35	6823893
36	6956180
37	7683145
38	8212180
39	8379000
40	8401292

Profit & Loss.csv

Day	Sales	Trading Profit	Operating Expense	Net Profit
35	24303924	8866269	2605390	6260279
36	24471890	8953446	2661675	6291771
37	25233785	9345165	2716605	6628560
38	25797345	9635457	2771130	6864327
39	26020982	9748900	2825655	6874707
40	26034115	9755787	2881080	6923245

Summary report based on the scenario 1



SCENARIO 2:

1. From the "Overheads.csv", Depreciation Expense is the highest overhead.
2. From the "Cash on Hand.csv" , the cash on day 36 is lower than day 35 and day 40 is lower than day 39.
3. From the "Profit & Loss.csv", net profit on day 38 is lower than day 37.

Overheads.csv

Category	Overheads
Salary Expense	28.77
Interest Expense	0.23
Rental Expense	20.64
Penalty Expense	12.88
Depreciation Expense	40.83
Human Resource Expense	16.66

Cash on Hand.csv

Cash On Hand	
Day	Cash On Hand
35	6956180
36	6823893
37	7683145
38	8212180
39	8401292
40	8379000

Profit & Loss.csv

Day	Sales	Trading Profit	Operating Expense	Net Profit
35	24303924	8866269	2605390	6260279
36	24471890	8953446	2661675	6291771
37	25797345	9635457	2771130	6864327
38	25233785	9345165	2716605	6628560
39	26020982	9748900	2825655	6874707
40	26034115	9755787	2881080	6923245

Summary report based on scenario 2

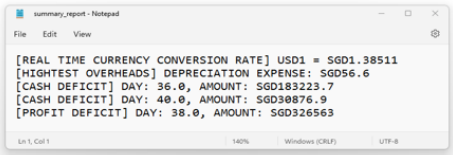


Figure 2.0 Modularizing the Program