The Brief

Automation Objectives

It is necessary to monitor business performance to identify the best performing periods so management can zoom in to analyze the success factors. Management can discuss and decide if the success factors can be repeated for greater success. At the same time, worst performing periods can also be analyzed to check if there are any failure factors that can be avoided in the future. This is a very tedious process that requires a lot of time and continuous effort. Automation will improve the efficiency of performance monitoring.

Python can be used to automate many office tasks. In this project, your team is required to develop a business automation program to automate the monitoring process:

1. Extract and summarise data from the finance dashboard in the final round of business stimulation game in MAB module. Refer to MAB module and MonsoonSim game on how to download the files. Your team is required to extract data **from day 11 to day 90**. You need to download the data a few times and combine the data together.

2. The automation will perform the tasks from the following csv files:
    a. Profit & Loss csv : The program will firstly compute the **difference** in the **net profit** column. If the net profit is always increasing, find out the day and amount the highest increment occurs. If the net profit is always decreasing, find out the day and amount the highest decrement occurs. If net profit fluctuates, list down all the days and amount when deficit occurs, and find out the top 3 highest deficit amount and the days it happened.
    b. Cash-On-Hand csv: The program will firstly compute the **difference** in Cash-on-Hand. If the cash-on-hand is always increasing, find out the day and amount the highest increment occurs. If the cash-on-hand is always decreasing, find out the day and amount the highest decrement occurs. If cash-on-hand fluctuates, list down all the days and amount when deficit occurs, and find out the top 3 highest deficit amount and the days it happened.
    c. Overheads csv: The program will find the **highest** overhead category.
    d. Write the computed amount from **a to c** to a text file and name it as summary_report.txt.

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

Your team will be required to analyze the best / worst performing day for success / failure factors as per the requirements in other modules, such as **SAPB**. For profit and loss, if it is always increasing / decreasing, your team is required to analyze the factors of highest surplus / deficit; when profit and loss fluctuates, your team is required to analyze the top 3 deficit amount and present the most meaningful analysis in the presentation. For details, please look through the group project requirements from other modules.

The same process applies to the cash-on-hand.

The Brief

Files and Project Directory

Organize you programs and csv files into the following folder structure: **(Your team is required to follow the structure and file names exactly. Otherwise, your team will be penalized.)**

Folder : project_group

```
| team_members.txt
| main.py
| cash_on_hand.py
| overheads.py
| profit_loss.py
| summary_report.txt
└── Folder: csv_reports
      Cash_on_Hand.csv
      Overheads.csv
      Profits_and_Loss.csv
```

Dedicate each python file to achieve specific tasks. For example, the cash_on_hand.py should only contain codes that compute the difference in Cash-on-Hand, while overheads.py should only contain codes that find the highest overhead category.

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

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.

Standard Criteria

The project will be evaluated based on:

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

The Brief

Bonus Marks

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

1. Collaborate on coding
2. Modularized the python files

How to Collaborate?

1. As this is a group project, you are expected to collaborate 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 Collaborate with GitHub.pdf)

Each member should be assigned to work on a specific part of the program. For example, a team member can work on the cash_on_hand.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 organize 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

The number of days is for illustration purposes here. Your team should download data from day 11 to day 90 for the project!

SCENARIO 1

1. Salary Expense is the highest overheads in “overheads.csv”
2. Each value on the current day is higher than the previous day in “cash_on_hand.csv” and “profit_and_loss.csv”

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

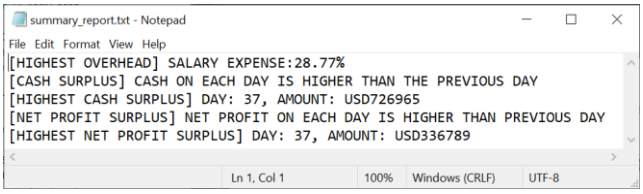
Cash_on_hand.csv

Cash On Hand	
Day	Cash On Hand
35	6823899
36	6956180
37	7683145
38	8212180
39	8379000
40	8401292

Profit_and_loss.csv

Day	Sales	Trading Profit	Operating Expense	Net Profit
35	24303924	8866269	2605990	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

Output: Summary_report.txt file content for scenario 1



SCENARIO 2

1. Depreciation Expense is the highest overheads in “overheads.csv”
2. Each value on the current day is lower than the previous day in “cash_on_hand.csv” and “profit_and_loss.csv”

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

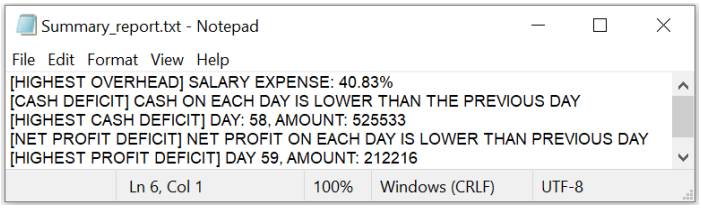
Cash_on_hand.csv

Cash on Hand	
Day	Cash On Hand
55	3142804
56	3117738
57	2995283
58	2469750
59	2367280
60	2026670

Profit_and_loss.csv

Profit and Loss				
Day	Sales	Trading Profit	Operating Expense	Net Profit
55	24581954	10961651	3103600	7858051
56	25243070	11352799	3489523	7855275
57	24439292	10868637	3025395	7843242
58	25080914	11252470	3411504	7832965
59	24915273	11154876	3334127	7620749
60	23996300	10579870	2946830	7533040

Output: Summary_report.txt file content for scenario 2



➔ Scenario 3 on next page

SCENARIO 3

1. Depreciation Expense is the highest overheads in “overheads.csv”
2. Each value on the current day can be higher or lower than the previous day in “cash_on_hand.csv” and “profit_and_loss.csv”

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
60	5578670
61	5274435
62	5254907
63	5281896
64	5315325
65	5401428
66	5031529
67	5126188
68	5153667
69	4888100
70	3144131

Profit_and_loss.csv

Profit and Loss				
Day	Sales	Trading Profit	Operating Expense	Net Profit
60	35397122	16013465	4876114	11129350
61	35497235	16073100	4967664	11097435
62	35598785	16134095	5044430	11081664
63	35692043	16195900	5119032	11088867
64	35791282	16256589	5188488	11060100
65	35942411	16351733	5260980	11082752
66	36094953	16449681	5346405	11075275
67	36248774	16552778	5413900	11130877
68	36402671	16656429	5483506	11164922
69	36554081	16759695	5557283	11194411
70	36707811	16864387	5627929	11228457

Output: Summary_report.txt file content for scenario 3

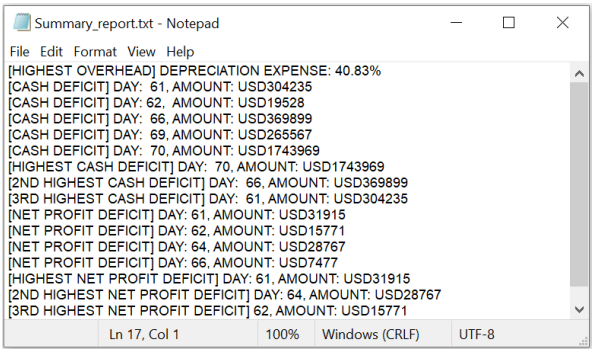


Figure 2.0 Modularizing the program

This is just an example. You are required to make necessary changes.