Assessment Information

In this assessment, you are required to complete the **interactive program (text-based)** to be installed in a Vending Machine in the University. This Vending Machine is used to dispense various products like **Tea, Coffee, Snacks, and Cold Drink** after coins are inserted into it.
In the following, this assessment will give a detailed description of the requirements for the "**automatic coin-operated drinks dispenser machine**" that provides **Tea, Coffee, Coke, and Orange Juice**.

Python version and software

In this task, you must use **python 3** and write all the code in a single **jupyter notebook** which means that there are **NO** additional .py files and all the **classes and/or functions MUST** be defined in that jupyter notebook you created for this project.

Functional requirements

In this assignment, you are required to build a vending machine that sells different kinds of drinks, i.e.: Tea, Coffee, Coke, and Orange Juice. In this program, you need to meet the following criteria.

In total, there are three kinds of requirements: User requirements (U-00X), System requirements (S-00X), and Data requirements (D-00X). The detailed requirements are prioritized as follows:

Table 1: Requirement priority level

Value	Rating	Description
1	Critical	This requirement is critical to the success of the project.
2	High	This requirement is high priority, but the project can be implemented at a bare minimum without this requirement
3	Medium	This requirement is somewhat important, as it provides some value but the project can proceed without it.
4	Low	This is a low priority requirement, or a "nice to have" feature, if time and cost allow it
5	Future	This requirement is out of scope for this project, and has been included here for a possible future release

Table2: Drinks dispenser machine requirements. Note that **Users** are the people who use the vending machine, **Customers** are the people who purchase the vending machine.

Req#	**Priority**	**Description**	**Rationale**	**Impacted stakeholders**	**Mark**
U-001	1	A new transaction shall be started	Users select **restart** to reset previous transactions and conditions to clear transaction history.	Users	4.00%
U-002	1	List of products must be displayed	Users select a product from the **list of products** on the display screen.	Users	4.00%
U-003	1	Cost of the selected item must be displayed	Users read and confirm by selecting **continue** option		4.00%
U-004	1	Coin must be inserted to get the product	A user can insert coins into the machine based on the indicated price. Note: In your program, you could use integers to represent the coins. For example: if the user wants to insert 50 cents, 20 cents, and 1 dollar, the inputs of the program could be '50, 20, 100' in cents.	Users	4.00%
S-002	1	Only readily items must be displayed as item list otherwise appropriate information to the user	E.g. coffee is not ready for sale	Users	4.00%
S-007	1	The system must dispense the selected item only if the amount coin is inserted unless it must rollback the transaction.		Users	4.00%
S-010	1	The system must accept **coins** of different in amount	E.g 10 cents, 20 cents, 50 cents, 1 dollars, 2 dollars	Users and Customers	4.00%
S-011	1	The system must compare item cost with entered coin			4.00%
S-012	1	The system must check the validity of the coin			4.00%
D-005	1	The system must store the coins and record all transaction			4.00%

U-005	2	The transaction shall be canceled as required	A user can select **Cancel** anytime during the transaction	Users	4.00%
U-007	2	Order shall be changed as required	A user can **change order** e.g. if s/he doesn't have enough coin to buy for the item they choose	Users	4.00%
S-001	2	The status of the machine shall be displayed	The system shall display status information whether it works or not	Users and Customers	4.00%
S-003	2	Alert message shall be printed automatically by the machine if any of ingredient is finished	E.g. if sugar is finished	Customers	4.00%
S-004	2	The system shall refund the coin if in need by the user before the item gets ready for dispense	Coin may be refunded when the user cancels the request or the coin they insert is not sufficient with appropriate information to the user	Users	4.00%
S-009	2	The system shall allow a user to select products coffee, tea, Coke, Juice, ...		Users	4.00%
D-006	2	The system need to provide statistical data based on the transactions			4.00%
U-006	3	The transaction shall be continued as required without restarting the whole steps	A user can select **Continue to Buy** option	Users	4.00%
S-005	3	The system shall display welcome and goodbye messages at the beginning and end of the transaction respectively.		Users and Customers	1.60%
S-006	3	The system must display waiting time as a message to the user in preparing the items	E.g. during boiling, adding ingredients and the like.	Users	1.60%
S-008	3	The system shall allow resetting operation for vending machine supplier.	E.g. for first time installation	Customers	2.40%
D-003	3	The system shall store Coffee and allow the user to mix sugar manually or automatically			2.40%

		by the machine			
					80.00%

Exception handling

Your program should try to avoid or catch any kinds of exceptions and errors, so the execution won't be interrupted and it won't enter abnormal states.

When handling exceptions related to user inputs, your system should return **meaningful error messages** so that the users can correct their inputs.

Documentation

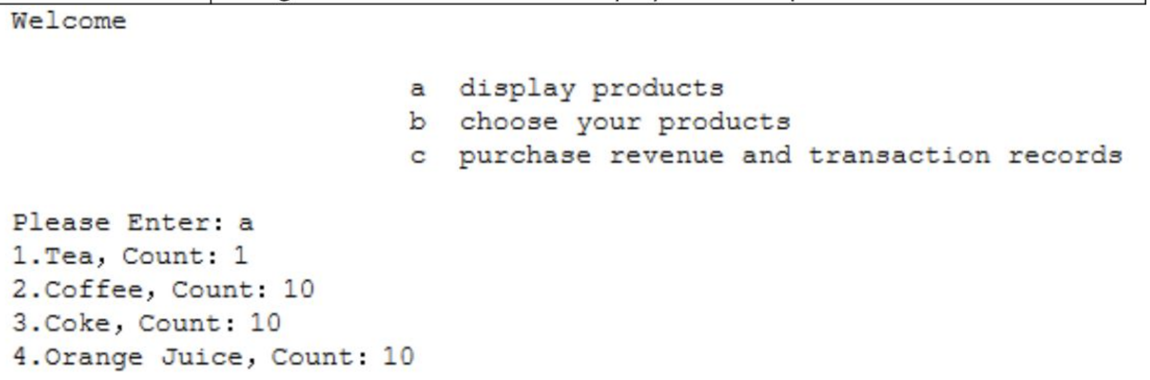
Precise and concise comments/documents of your code are essential as part of assessment criteria. Do Not include things that are irrelevant in your code as that will reduce your code readability.

In addition, you need to submit a **final report** which covers the following points (but not limited to):

1. Describing the **high-level design** of your programs
2. Designing the **user cases** to address **EACH** requirements mentioned above. In this section, you need to capture screenshots of the results while executing your program. Our accessor will mark the functionality part mainly based on the user cases and the captured results from executing your program. Below is the example of one user case

Note: sufficient evidence to prove the implementation of the requirements should be provided, otherwise marks will be deducted.

Name	UC002, Display the products
Actor(s)	User
Goal	The goal of this use case is to display the list of products



```
Welcome

                         a  display products
                         b  choose your products
                         c  purchase revenue and transaction records

Please Enter: a
1.Tea, Count: 1
2.Coffee, Count: 10
3.Coke, Count: 10
4.Orange Juice, Count: 10
```

Grading Rubric

Parts	**mark allocation**
Functionality	80%
Code quality & architect	10%
final report	10%

Note that: the detailed mark allocation is shown in the functional requirement section.

Penalties

- Late submission: -5% per day late, no submission accepted after 4 days.

For further details on late submission, **read the policy here**.

Academic Integrity: Plagiarism and Collusion

Plagiarism Plagiarism means to take and use another person's ideas and or manner of expressing them and to pass them or as your own by failing to give appropriate acknowledgement.

This includes materials sourced from the Internet, staff, other students, and from published and unpublished works.

Collusion Collusion means unauthorised collaboration on assessable work (written, oral, or practical) with other people. This occurs when you present group work as your own or as the work of another person. Collusion may be with another Monash student or with people or students external to the University. This applies to work assessed by Monash or another university.

It is your responsibility to make yourself familiar with the University's policies and procedures in the event of suspected breaches of academic integrity. (Note: Students will be asked to attend an interview should such a situation is detected.)
The University's policies are available at: http://www.monash.edu/students/academic/policies/academic-integrity

Submission

- Submit a folder named with your student ID. This directory should contain **vending_machine.ipynb**, **system files and final_report.pdf**.
- Write all the code and documentation in **vending_machine.ipynb**
- Do not include any unnecessary file in this folder
- Zip the containing folder with the same name (ie <student_id>.zip) and upload it