CS4051NI/CC4059NI Fundamentals of Computing

70% Individual Coursework

2024/25 Spring

Student Name: Kesang Wangmo Lama

London Met ID: 24046590

College ID: NP01AI4A240002

Assignment Due Date: Wednesday, May 14, 2025

Assignment Submission Date: Wednesday, May 14, 2025

Word Count: 4848

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

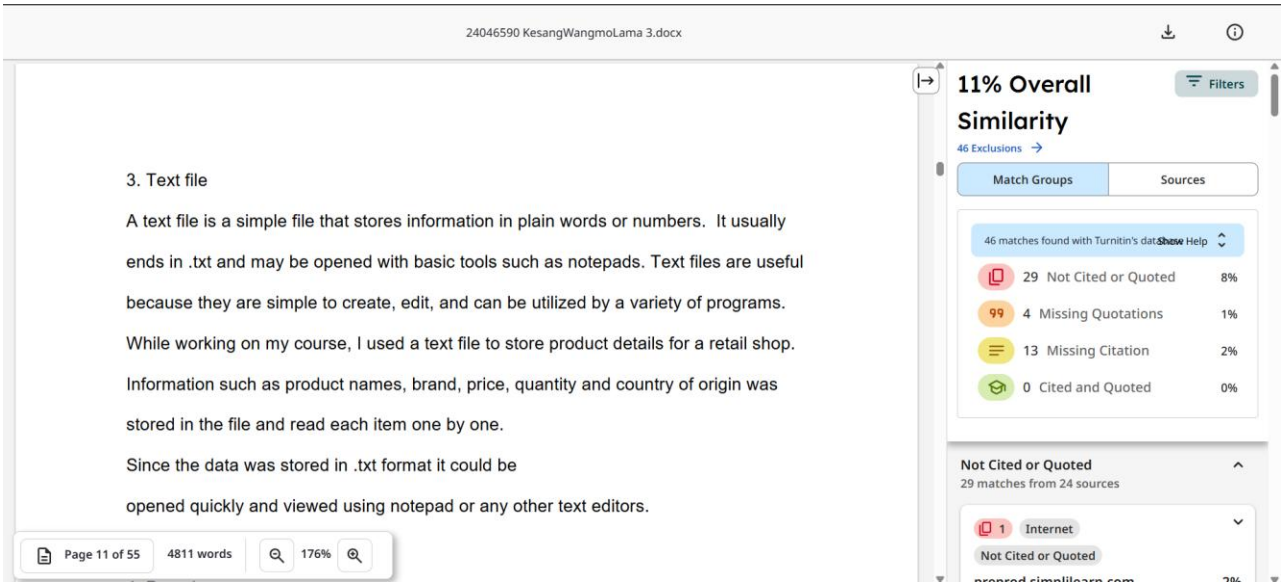

24046590 KesangWangmoLama 3.docx
3. Text file
A text file is a simple file that stores information in plain words or numbers. It usually ends in .txt and may be opened with basic tools such as notepads. Text files are useful because they are simple to create, edit, and can be utilized by a variety of programs. While working on my course, I used a text file to store product details for a retail shop. Information such as product names, brand, price, quantity and country of origin was stored in the file and read each item one by one.
Since the data was stored in .txt format it could be
opened quickly and viewed using notepad or any other text editors.
Page 11 of 55
4811 words
176%
11% Overall Similarity
Filters
46 Exclusions
Match Groups
Sources
46 matches found with Turnitin's database
Show Help
29 Not Cited or Quoted 8%
4 Missing Quotations 1%
13 Missing Citation 2%
0 Cited and Quoted 0%
Not Cited or Quoted
29 matches from 24 sources
1 Internet
Not Cited or Quoted

Table of Contents

Table of figures

Table of Tables

1. INTRODUCTION

About the Project

The project that I am working on focuses on developing a Product wholesale system developed for a retail store named WeCare. WeCare, like any other retail store, needs precise control over sales transactions, including the application of promotional discounts which may have an impact on inventory and stock control accounting. At the most basic level inventory control consists of reading the stock record from a file, processing customer purchases, updating inventory, and preparing detailed invoices. One significant characteristics of the developed system is its built in promo logic, specifically a “buy three, get one free” policy which triggers during the check out process. Payment processing features accurate calculations of the total amount to be paid, incorporating the markup percentage, but excludes free items from billing. There is also a provision for restocking whereby staff can change the amount and cost price of an item and an invoice can be generated for the suppliers. Product information is stored in an efficient manner by using lists and dictionaries provided by the Python programming language. The system integrates all critical features to meet the operational requirements with limited manual intervention to enhance accuracy and streamline workflow, ease the rick of operational mistakes, and automate inventory control.

In summary, the Product Wholesale System for WeCare was created to handle critical retail processes such as inventory tracking, sales processing, promotional handling, and restocking in an efficient and automated manner. By using Python's list and dictionary structures, the system ensures ordered data management and easy operation. Features such as the "buy three, get one free" promotion and automated invoice production decrease manual errors while saving time. Overall, this system intends to increase retail staff productivity, accuracy, and convenience of use, allowing WeCare to better manage its operations.

Aim and Objectives

The goal of this particular project is to create a comprehensive automated wholesale management system facilitating the retailers inventory oversight, correct sales and transactions, implement promotional pricing strategies, and payroll with zero errors and stock refresh in real rime leading to increases productivity and operational efficiency.

The main objectives of the project are:

1. To create a new approach pertaining to tracking stock levels of products in inventory by reading, showcasing and modifying the levels in real time as sales and restocking actions are performed.

2. To enable accurate billing of the total payable amount, which has a marked up order value on the cost price while classifying free of charge goods as non invoiced items.

3. To automatically create organized records for sales and restocking, including product details, customer or supplier names, and the total amount.

4. To introduce and active logic for promotional sales that grants the “buy three, get one free” option automatically without any manual control during checkouts.

5. To reduce human errors, automate repetitive processes like stock updates, total price calculation, and free item tracking.

List of tools and technologies used

1. Python programming Language:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its highlevel built-in data structures, together with dynamic typing and dynamic binding, making it ideal for Rapid Application Development and as a scripting or glue language for connecting existing components. Python's concise, easy-tolearn syntax promotes readability, lowering the cost of software maintenance. Python provides modules and packages, promoting program modularity and code reuse (AS, 2024).

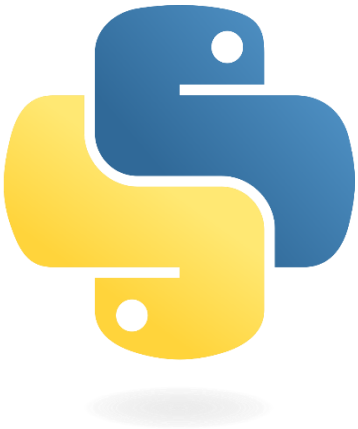


Figure 1. python

2. IDLE

IDLE which stands for Integrated Development and Learning Environment is a built in development environment that comes standard with the python distribution. It is a great tool for beginner like us who are just starting to learn Python programming. It has an interactive shell where we can rapidly test commands and see the results right away. For writing and storing Python applications, the idle additionally features a script editor. My coding

Figure 2.IDLE

experience was made easy, and uncomplicated by the IDLE's useful features, which included syntax highlighting, automatic indentation, and basic debugging (Jaishree, 2025)

3. Text file

A text file is a simple file that stores information in plain words or numbers. It usually ends in .txt and may be opened with basic tools such as notepads. Text files are useful because they are simple to create, edit, and can be utilized by a variety of programs. While working on my course, I used a text file to store product details for a retail shop. Information such as product names, brand, price, quantity and country of origin was stored in the file and read each item one by one.

Figure 3.txt file

Since the data was stored in .txt format it could be opened quickly and viewed using notepad or any other text editors.

4. Draw.io

Draw.io is an exclusive software for creating diagrams and charts. The software allows you to use an automatic layout function or create a bespoke layout. They provide a huge selection of shapes and hundreds of visual elements to help you create a unique diagram or chart. The drag-and-drop tool makes it easy to create a visually appealing diagram or chart. (draw.io, 2024)

Figure 4.draw.io

2. Discussion and Analysis

Algorithm: Product Wholesale System

Step 1: Start
Step 2: Display inventory table
Step 3: Show option prompt
Step 4: if user inputs 1
Step 5: Show stock and available products
Step 6: Ask for the customer's name and contact
Step 7: input product ID and quantity
Step 8: Validate ID and quantity
    If invalid, go to step 7
Step 9: Apply the "But 3 get one free offer"
Step 10: Ask if shipping is required
    If yes, add shipping fee
    Else, proceed without it
Step 11: Generate and save invoice to a .txt file
Step 12: update and save stock to file
Step 13: Else if suer inputs 2
Step 14: Display product list
Step 15: input product ID. Quantity and new price(optional)
Step 16: Validate and update stock
Step 17: Save updated stock to file
Step 18: Else if user inputs 3
Step 19: End the program

Explanation:

The program begins by displaying a list of available inventory goods, followed by a popup that asks the user to select between purchasing, replenishing, or leaving the application.

If the user chooses the purchase option, they are prompted to complete customer information before selecting products by entering product IDs and requested quantities. The system checks for legitimate entries and applies a "buy three, get one free" promotion when appropriate. It then asks if the customer wants shipment, charges a shipping fee if necessary, and calculates the final total. It then generates a thorough invoice and saves it to a text file, updating the stock data.

If the user decides to restore the system allows them to update inventory by selecting a product and inputting the quantity to be added, with the option to adjust the product's pricing. The is then validated and saved back into the file. The software will run in a look unitl the user picks the exit option.

Flowchart

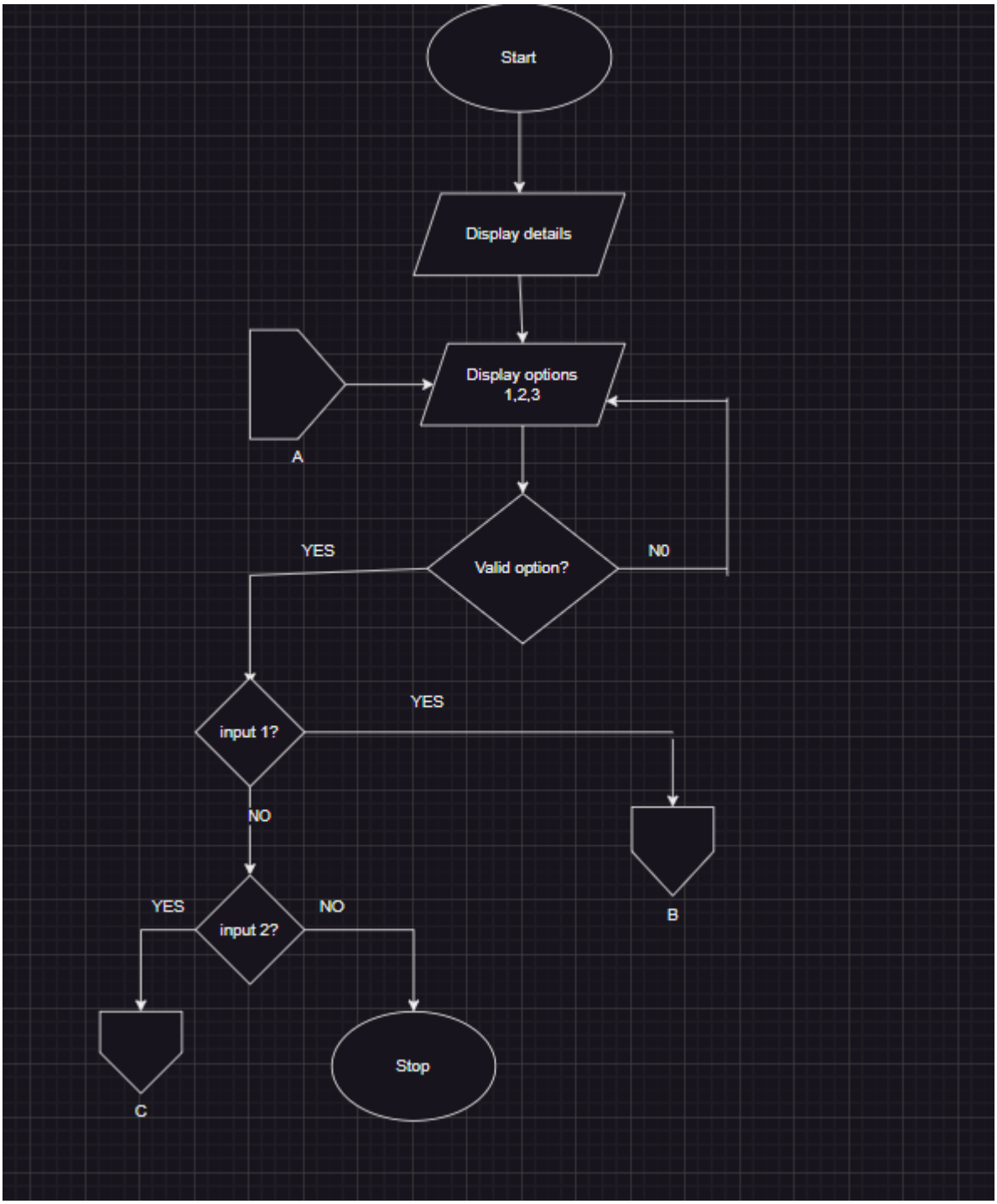


Figure 5.flowchart 1

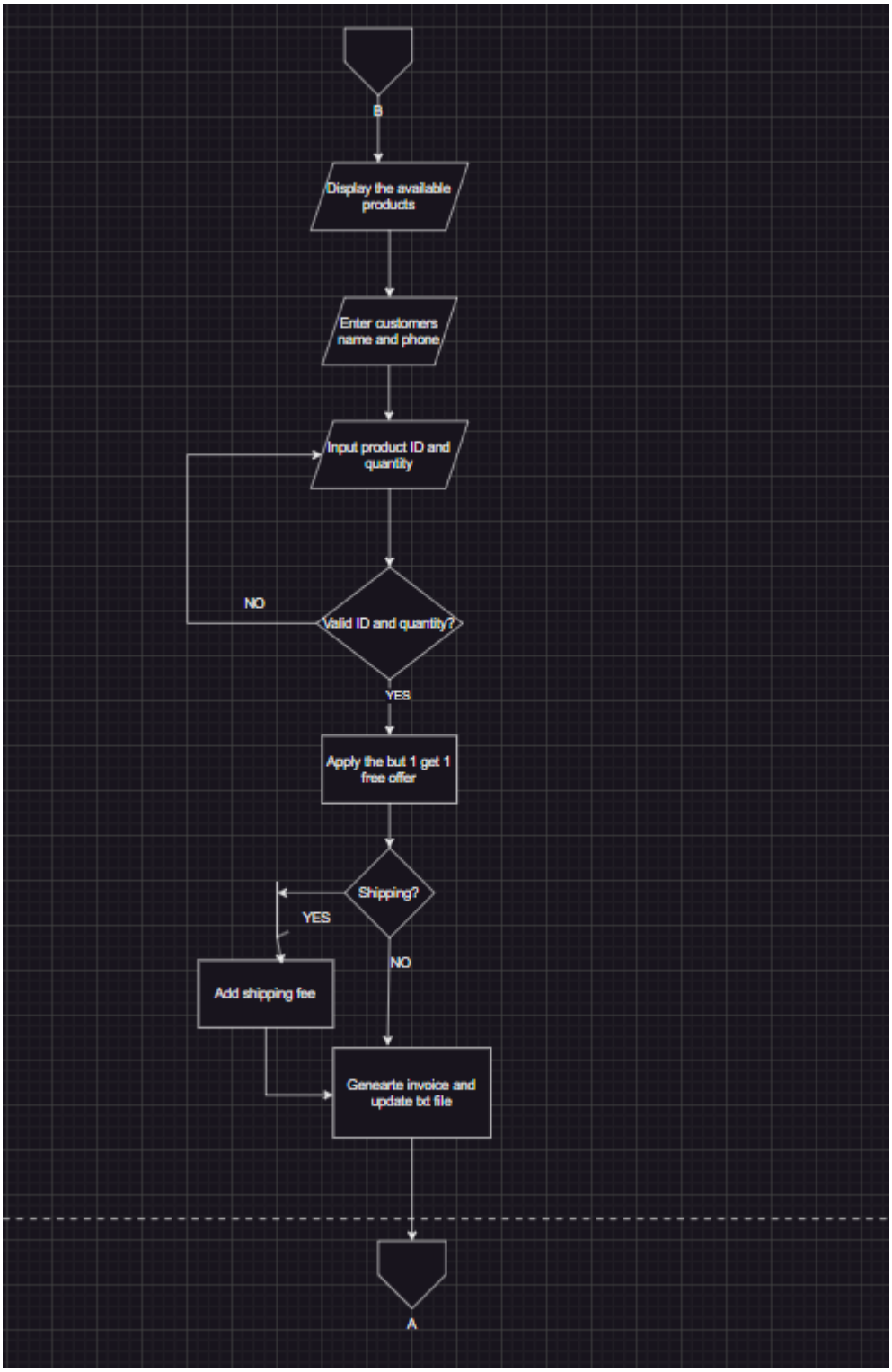


Figure 6.flowchart 2

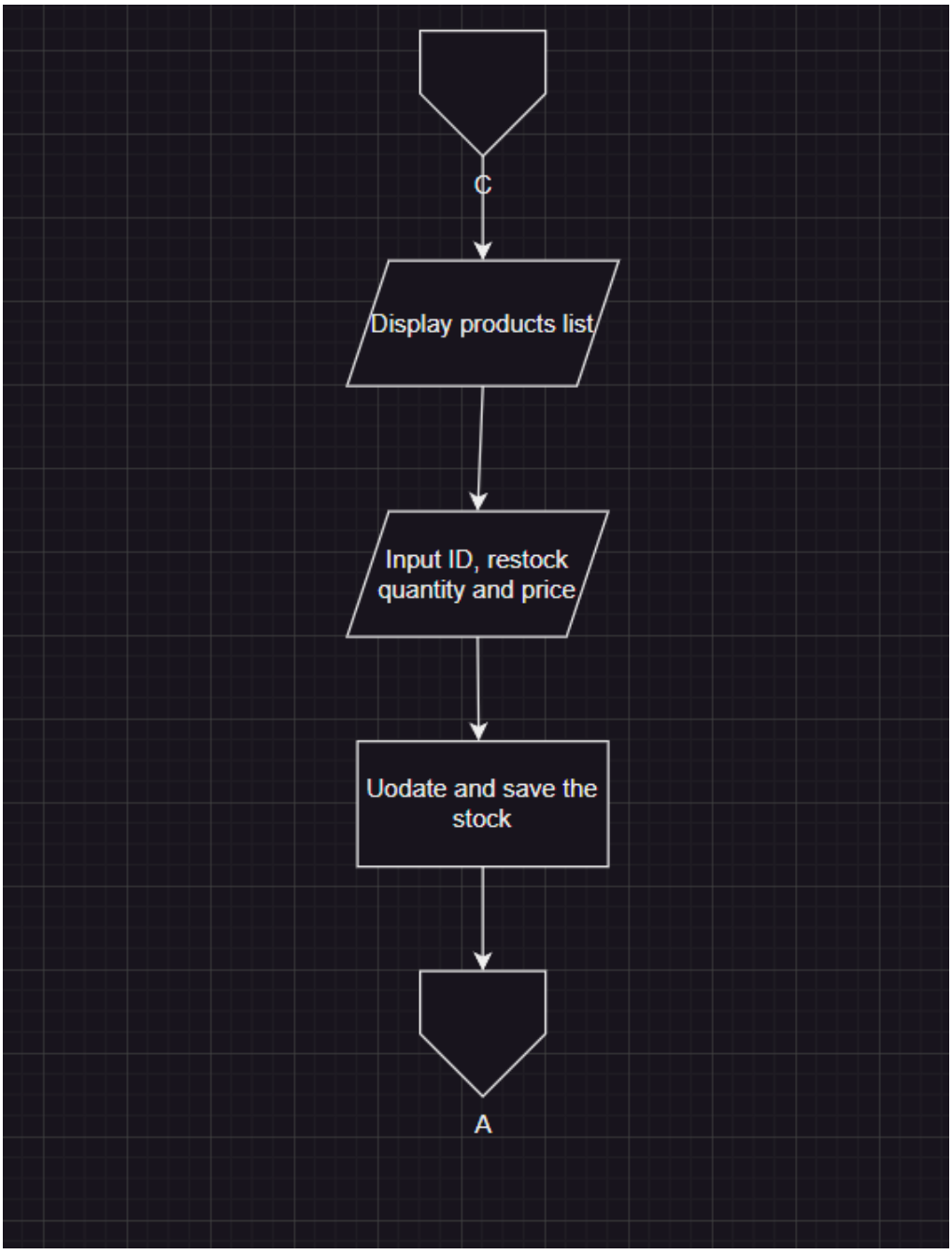


Figure 7.flowchart 3

Explanation

This flowchart shows the logic behind an inventory management system. It begins by displaying available products and giving the user three options: purchase , restock, or exit. Depending on the user's preferences, the system either handles multiple product sales with invoice creation and delivery options, or restocks products with updated quantities and pricing. Connectors are used to control the flow between sections and loop back to the main menu. The procedure continues until the user picks the exit option, after which the software terminates.

Pseudocode

1. Read.py

```
ALGORITHM GET_DATA
    INITIALIZE PRODUCT_DATA AS AN EMPTY DICTIONARY
    OPEN FILE "DATA.TXT" IN READ MODE
    READ ALL LINES FROM THE FILE AND STORE IN FF
    SET ITEM_ID = 1

    FOR EACH DATA IN FF DO
        REMOVE NEWLINE CHARACTER FROM DATA
        SPLIT DATA BY COMMA AND STORE IN LIST
        ASSIGN LIST TO PRODUCT_DATA[ITEM_ID]
        INCREMENT ITEM_ID BY 1
    END FOR

    CLOSE THE FILE
    RETURN PRODUCT_DATA
END ALGORITHM
```

2. write.py

```
ALGORITHM GENERATE_INVOICE
    INPUT FILE_PATH, NAME, PHONE_NUMBER, DATETIME_NOW, ITEMS,
SHIPPING_COST, GRAND_TOTAL

    OPEN FILE_PATH IN APPEND MODE
```

```
    WRITE HEADER AND COMPANY DETAILS
    WRITE "CUSTOMER DETAILS" SECTION
    WRITE NAME, PHONE_NUMBER, DATETIME_NOW

    WRITE "PURCHASE DETAIL" HEADER
    WRITE COLUMN TITLES: ITEM NAME, QUANTITY, FREE, UNIT PRICE, TOTAL

    FOR EACH ITEM IN ITEMS DO
        WRITE ITEM NAME, QUANTITY, FREE QTY, UNIT PRICE, TOTAL PRICE
    END FOR

    IF SHIPPING_COST > 0 THEN
        WRITE SHIPPING COST
    END IF

    WRITE GRAND TOTAL
    CLOSE FILE

END ALGORITHM

ALGORITHM RESTOCK_INVOICE
    INPUT file_path, restock_records, datetime_now

    OPEN file_path IN append mode
    WRITE store name and restock header
    WRITE date and time of restocking

    FOR EACH record IN restock_records DO
        WRITE product name, quantity added, new total quantity, updated price
    END FOR

    CLOSE the file
```

END RESTOCK_INVOICE

3. operations.py

```
ALGORITHM SHOW_PRODUCTS
  INPUT PRODUCT_DATA

  PRINT "AVAILABLE PRODUCTS"
  FOR EACH KEY, VALUE IN PRODUCT_DATA DO
    PRINT PRODUCT DETAILS WITH QUANTITY AND COST PRICE
  END FOR

END ALGORITHM


ALGORITHM GET_MENU
  LOOP
    TRY TO INPUT CHOICE AS INTEGER
    IF CHOICE IS 1 OR 2 OR 3 THEN
      RETURN CHOICE
    ELSE
      PRINT "PLEASE CHOOSE 1, 2, OR 3"
    END IF
  IF ERROR OCCURS THEN
    PRINT "INVALID INPUT. PLEASE ENTER A NUMBER"
  END LOOP
END ALGORITHM



ALGORITHM GET_PRODUCT_ID
```

```
    INPUT PRODUCT_DATA

    LOOP
      TRY TO INPUT PRODUCT_ID AS INTEGER
      IF PRODUCT_ID IS BETWEEN 1 AND LENGTH OF PRODUCT_DATA THEN
        RETURN PRODUCT_ID
      ELSE
        PRINT "PRODUCT NOT AVAILABLE. PLEASE TRY AGAIN"
      END IF
    IF ERROR OCCURS THEN
      PRINT "PLEASE ENTER A VALID ID"
    END LOOP
  END ALGORITHM

  ALGORITHM GET_PRODUCT_ID
    INPUT PRODUCT_DATA

    LOOP
      TRY TO INPUT PRODUCT_ID AS INTEGER
      IF PRODUCT_ID IS BETWEEN 1 AND LENGTH OF PRODUCT_DATA THEN
        RETURN PRODUCT_ID
      ELSE
        PRINT "PRODUCT NOT AVAILABLE. PLEASE TRY AGAIN"
      END IF
    IF ERROR OCCURS THEN
      PRINT "PLEASE ENTER A VALID ID"
    END LOOP
  END ALGORITHM

  ALGORITHM GET_PURCHASE_QTY
    INPUT AVAILABLE_QTY
```

```
    LOOP
      TRY TO INPUT QUANTITY AS INTEGER
      CALCULATE FREE = QUANTITY DIVIDED BY 3
      CALCULATE TOTAL_NEEDED = QUANTITY + FREE

      IF QUANTITY > 0 AND TOTAL_NEEDED <= AVAILABLE_QTY THEN
        RETURN QUANTITY, FREE
      ELSE
        PRINT "THE QUANTITY IS NOT AVAILABLE"
      END IF
    IF ERROR OCCURS THEN
      PRINT "INVALID QUANTITY"
    END LOOP
  END ALGORITHM

  ALGORITHM GET_RESTOCK_QTY

    LOOP
      TRY TO INPUT QTY AS INTEGER
      IF QTY > 0 THEN
        RETURN QTY
      ELSE
        PRINT "QUANTITY MUST BE GREATER THAN 0"
      END IF
    IF ERROR OCCURS THEN
      PRINT "INVALID INPUT. ENTER A VALID NUMBER"
    END LOOP

  END ALGORITHM

  ALGORITHM SAVE_DATA
    INPUT PRODUCT_DATA
```

```
    OPEN "DATA.TXT" IN WRITE MODE
    FOR EACH VALUE IN PRODUCT_DATA DO
      WRITE VALUES JOINED BY COMMAS TO FILE
    END FOR
    CLOSE FILE

END ALGORITHM
```

4.Main.py

```
    ALGORITHM MAIN_PROGRAM
      START
      WHILE main_loop IS TRUE DO
        CALL LOAD_DATA AND STORE IN product_data

        DISPLAY ALL product_data WITH HEADERS
        DISPLAY MENU OPTIONS (1. Purchase, 2. Restock, 3. Exit)

        GET choice USING get_menu_option()

        IF choice = 1 THEN
          CALL show_products(product_data)
          GET name, phone_number FROM USER

          INITIALIZE purchased_items, total, shipping_cost, grand_total

          WHILE TRUE DO
            CALL get_product_id(product_data) AND STORE IN product_id
            GET qty_available FROM product_data[product_id][2]
            CALL get_purchase_qty(qty_available) AND STORE quantity, free_qty

            CALCULATE unit_price, total_price
```

```
            UPDATE STOCK: product_data[product_id][2] = qty_available - quantity -
    free_qty
                ADD purchase detail TO purchased_items
                INCREMENT total BY total_price

                ASK USER IF THEY WANT TO BUY MORE
                IF RESPONSE IS NOT 'Y' THEN BREAK
            END WHILE

            ASK IF SHIPPING REQUIRED
            IF YES THEN ADD 500 TO shipping_cost

            CALCULATE grand_total = total + shipping_cost
            CALL save_data(product_data)

            DISPLAY INVOICE ON SCREEN
            CREATE filename BASED ON TIMESTAMP
            CALL generate_invoice(filename, name, phone_number, date_time,
    purchased_items, shipping_cost, grand_total)

        ELSE IF choice = 2 THEN
            DISPLAY "Restocking the products"
            INITIALIZE restock_records

            WHILE TRUE DO
                CALL show_products(product_data)
                CALL get_product_id(product_data) AND STORE IN restock_id
                CALL get_restock_qty() AND STORE IN qty
                GET cost INPUT FROM USER

                UPDATE product_data QUANTITY AND PRICE
                CALL save_data(product_data)
```

```
            APPEND RESTOCK DETAILS TO restock_records

            ASK USER IF THEY WANT TO RESTOCK ANOTHER
            IF RESPONSE IS NOT 'Y' THEN BREAK
          END WHILE

          CREATE filename BASED ON TIMESTAMP
          CALL restock_invoice(filename, restock_records, today_date_and_time)

          DISPLAY RESTOCK SUMMARY ON SCREEN

        ELSE IF choice = 3 THEN
          SET main_loop TO FALSE
          DISPLAY EXIT MESSAGE
        END IF
      END WHILE
    END MAIN_PROGRAM
```

Data and Structures

In today's environment, the importance of data has grown rapidly. To successfully administer an organization, all obtained data must be preserved. Data structure is just a structure that can hold some data together. In other words, it's used to hold a group of similar data. Each programming language has its own data structure. Python includes four built-in data structures: lists, tuples, dictionaries, and sets. The simplest basic data structure in Python is a sequence. Each element in the series is assigned an index. The first index is 0, the second index is 1, and so on.

The description of the data structures and how they are used are as follows:

1. Primitive Data Type

The most basic form of data storage which only represents a single value is primitive data storage.

- Integer

Integer represents whole numbers both positive and negative without any decimal points. For example, age = 18.

- Float

Float refers to the real numbers with decimal points. Float is used for precise calculations. Example, price = 59.99

- Character

Characters are represented as strings which are sequences of Unicode characters. For example, letter = ‘A’

- Boolean

Boolean represents one of wo values ‘true’ or ‘false’, it is commonly used for conditional statements. For example, is_valid = True

2. Non-Primitive Data Type

Non-Primitive Data structure are more complicated than primitive data structures and can store multiple values.

- List: A list in python is a built-in dynamic sized array. It is used to store multiple items in a single variable. List items are ordered, changeable and allow duplicate values. The items in list are indexed from [0] to [len(list)-1]. It is also ordered, changeable and allows duplicate values (w3schools, n.d.).

Example from my code:

```
for key, value in product_data.items():
    print(key, ":", value)
```

The line print(key, ":", value) outputs each product ID along with its associated list of details. This shows that product_data is a dictionary of lists.

```
['Glycolicserum', 'Ordinary', '216', '500', 'USA']
['Night_cream', 'Cetaphil', '52', '280', 'Switzerland']
['Sunscreen', 'BeautyofJoseon', '141', '700', 'Korea']
['Eyecream', 'Centella', '0', '300', 'China']
['Moisturizer', 'Mamaearth', '344', '500', 'India']
['FaceCleanser', 'HadaLabo', '700', '900', 'Japan']
['lipbalm_spf', 'Dermaco', '340', '1200', 'India']
```

- Dictionary: A dictionary in python is a data structure that stores the value in key: value pairs. Values in a dictionary can be of any data type and can be duplicated, whereas keys can’t be repeated and must be immutable

(w3schools, n.d.). In this project, we have used as the main storage structure for all products in the system. Each product in the system has a unique ID (item_id), which serves as the key. The value associated with that key is the list of the product qualities described before. This setup enables the system to rapidly loop up a product by ID, update its inventory, display its details, or remove it is necessary.

Example from my code:

```
product_data = load_data()
```

product_data is a dictionary where each key is an item ID and each value is a list containing product attributes such as name, brand, quantity, price, and country.

```
1 : ['Glycolicserum', 'Ordinary', '216', '500', 'USA']
2 : ['Night_cream', 'Cetaphil', '52', '280', 'Switzerland']
3 : ['Sunscreen', 'BeautyofJoseon', '141', '700', 'Korea']
4 : ['Eyecream', 'Centella', '0', '300', 'China']
5 : ['Moisturizer', 'Mamaearth', '344', '500', 'India']
6 : ['FaceCleanser', 'HadaLabo', '700', '900', 'Japan']
7 : ['lipbalm_spf', 'Dermaco', '340', '1200', 'India']
```

- String: String in python are surrounded by either single quotation marks, or double quotation marks. It can simply be created using a enclosing a character inside quotes.

  Example from my code

```
file = open("data.txt", "r")
ff = file.readlines()
item_id = 1
for data in ff:
    data = data.replace("\n", "").split(",")
    product_data[item_id] = data
    item_id += 1
file.close()
```

In my code, each line from data.txt is read as a string. I used replace("\n", "") to remove the newline, then split(",") to convert the string into a list. This is a clear example of how string manipulation is used in my project.

- Set: Set is a data type in python used to store several items in a single variable. It is one of the four built-in data types (List, Dictionary, Tuple, and Set) having qualities and usage different from the other three. It is a collection that is written with curly brackets and is both unindexed and unordered.
  A set is mutable, i.e., we can remove or add elements to it. Set in python is similar to mathematical sets, and operations like intersection, union, symmetric difference, and more can be applied (AS, 2024).
  Example:

```
# Creating a set
fruits = {"apple", "banana", "orange"}
# Adding an item
fruits.add("mango")
 # Removing an item fruits.remove("banana")
 print(fruits)
```

- Tuples: A Python tuple is a collection of items or values. Python tuple can be created by specifying comma separated values inside of parentheses ( ). Values inside of a tuple cannot be modified once created. Python tuples follow the idea of packing and unpacking values i.e. while creating a tuple

parentheses ( ) are optional and just providing a comma-separated value will create a tuple as well (also known as packing). Similarly, when trying to access each value in the tuple, the values can be assigned to individual variables (also called unpacking) (brain station, 2025).

Example:

```
# Creating a tuple
person = ("Alice", 25, "Canada")


# Accessing elements using indexing
print(person[0])  # Output: Alice
print(person[1])  # Output: 25
```

3. Program

The program was designed to manage products in a small shop. It offers the user three choices:

1. Buy a product
2. Restock a product
3. Exit the program

If the user picks the first option, they need to submit their name and contact information, select one or more products, and the system checks the number and see if its valid or not and gives error message accordingly and then it applies a “buy 3 get 1 free” deal, inquires about delivery and ultimately prints and stores a comprehensive invoice in a txt file.

Step wise execution of the program:

```
1 : ['Glycolicserum', 'Ordinary', '16', '500', 'USA']
2 : ['Night_cream', 'Cetaphil', '52', '280', 'Switzerland']
3 : ['Sunscreen', 'BeautyofJoseon', '141', '700', 'Korea']
4 : ['Eyecream', 'Centella', '0', '300', 'China']
5 : ['Moisturizer', 'Mamaearth', '344', '500', 'India']
6 : ['FaceCleanser', 'HadaLabo', '700', '900', 'Japan']
7 : ['lipbalm_spf', 'Dermaco', '340', '1200', 'India']
-------------------------------------------------------------------------------
ID       Product         brand           qty     price   country
-------------------------------------------------------------------------------
1       Glycolicserum   Ordinary        16      500     USA
2       Night_cream     Cetaphil        52      280     Switzerland
3       Sunscreen       BeautyofJoseon  141     700     Korea
4       Eyecream        Centella        0       300     China
5       Moisturizer     Mamaearth       344     500     India
6       FaceCleanser    HadaLabo        700     900     Japan
7       lipbalm_spf     Dermaco 340     1200    India
-------------------------------------------------------------------------------
 1. Purchase
 2. Restock a product
 3. Exit the application
Enter the option to continue:
```

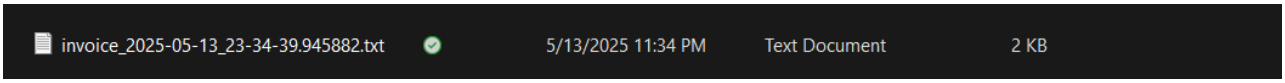
This part of the program displays the current inventory loaded from a file. Each product is listed with its ID, name, brand, quantity, price, and country of origin. After displaying the stock, the user is given three options: purchase a product, restock a product, or exit the application.

```
Enter the option to continue: 1



Available Products:
1. Glycolicserum (Quantity: 16, Cost Price: 500)
2. Night_cream (Quantity: 52, Cost Price: 280)
3. Sunscreen (Quantity: 141, Cost Price: 700)
4. Eyecream (Quantity: 0, Cost Price: 300)
5. Moisturizer (Quantity: 344, Cost Price: 500)
6. FaceCleanser (Quantity: 700, Cost Price: 900)
7. lipbalm_spf (Quantity: 340, Cost Price: 1200)
Enter your details to generate the bill:
Please enter the name of the customer: kesang
Please enter the phone number of the customer: 9872633493
Enter the ID of the product: 3
please provide the quantity you want to buy:10

Dear kesang you received 3 items for free as part of the offer.
Do you want to buy another item? (Y/N): y
Enter the ID of the product: 7
please provide the quantity you want to buy:500
The quantity is not available
please provide the quantity you want to buy:100

Dear kesang you received 33 items for free as part of the offer.
Do you want to buy another item? (Y/N): n
Do you want your products to be shipped? (Y/N): y
```

This shows the process of purchasing products in the system. The customer selects items, enters the quantity, and receives free items as part of a promotional offer. If the quantity requested is too high, the system asks again, and at the end, it checks whether shipping is needed.

```
Enter the option to continue: 2


Restocking the products

Available Products:
1. Glycolicserum (Quantity: 16, Cost Price: 500)
2. Night_cream (Quantity: 52, Cost Price: 280)
3. Sunscreen (Quantity: 128, Cost Price: 700)
4. Eyecream (Quantity: 0, Cost Price: 300)
5. Moisturizer (Quantity: 344, Cost Price: 500)
6. FaceCleanser (Quantity: 700, Cost Price: 900)
7. lipbalm_spf (Quantity: 207, Cost Price: 1200)
Enter the ID of the product: 4
Enter quantity to add: 100
Enter new price: 400
Product restocked successfully!
Do you want to restock another product?(Y/N):n

1 : ['Glycolicserum', 'Ordinary', '16', '500', 'USA']
2 : ['Night_cream', 'Cetaphil', '52', '280', 'Switzerland']
3 : ['Sunscreen', 'BeautyofJoseon', '128', '700', 'Korea']
4 : ['Eyecream', 'Centella', '100', '400', 'China']
5 : ['Moisturizer', 'Mamaearth', '344', '500', 'India']
6 : ['FaceCleanser', 'HadaLabo', '700', '900', 'Japan']
7 : ['lipbalm_spf', 'Dermaco', '207', '1200', 'India']
```

This shows the restocking process in the system. The user selects a product by its ID, adds more quantity, optionally updates the price, and the system confirms the update. After restocking, the updated product list reflects the changes made.



Then a .txt file was created

```
==========================================================================================================
                        We Care
                Boudha, Kathmandu | Phone No: 9841277311
----------------------------------------------------------------------
Customer Details:
----------------------------------------------------------------------
Name of the Customer: kesang
Contact number: 9872633493
Date and time of purchase: 2025-05-13 23:34:39.945882
----------------------------------------------------------------------

Purchase Detail:
----------------------------------------------------------------------------------------------------------------------
Item Name       Quantity        Free    Unit Price      Total
----------------------------------------------------------------------------------------------------------------------
Sunscreen       10              3               1400            $14000
lipbalm_spf     100             33              2400            $240000
----------------------------------------------------------------------------------------------------------------------
Shipping Cost: $500
Grand Total: $254500
```

This is the bill generated inside the txt file created earlier.

```
1 : ['Glycolicserum', 'Ordinary', '16', '500', 'USA']
2 : ['Night_cream', 'Cetaphil', '52', '280', 'Switzerland']
3 : ['Sunscreen', 'BeautyofJoseon', '128', '700', 'Korea']
4 : ['Eyecream', 'Centella', '100', '400', 'China']
5 : ['Moisturizer', 'Mamaearth', '344', '500', 'India']
6 : ['FaceCleanser', 'HadaLabo', '700', '900', 'Japan']
7 : ['lipbalm_spf', 'Dermaco', '207', '1200', 'India']
------------------------------------------------------------------------------
ID       Product          brand            qty     price   country
------------------------------------------------------------------------------
1       Glycolicserum   Ordinary        16      500     USA
2       Night_cream     Cetaphil        52      280     Switzerland
3       Sunscreen       BeautyofJoseon  128     700     Korea
4       Eyecream        Centella        100     400     China
5       Moisturizer     Mamaearth       344     500     India
6       FaceCleanser    HadaLabo        700     900     Japan
7       lipbalm_spf     Dermaco 207     1200    India
------------------------------------------------------------------------------
 1. Purchase
 2. Restock a product
 3. Exit the application
Enter the option to continue: 3
```

This shows the termination of the program after selecting the option 3 that is the exit option.

4. Testing

4.1. Test 1

Objectives	Test the implementation of try and except to handle invalid input
Action	Enter a invalid input when asked to enter a product ID
Expected Result	Program should display an error message
Actual Result	Program displayed “invalid input”
Conclusion	The error handling works correctly and the test was successful.

Table 1 test 1

```
1 : ['Glycolicserum', 'Ordinary', '16', '500', 'USA']
2 : ['Night_cream', 'Cetaphil', '52', '280', 'Switzerland']
3 : ['Sunscreen', 'BeautyofJoseon', '128', '700', 'Korea']
4 : ['Eyecream', 'Centella', '100', '400', 'China']
5 : ['Moisturizer', 'Mamaearth', '344', '500', 'India']
6 : ['FaceCleanser', 'HadaLabo', '700', '900', 'Japan']
7 : ['lipbalm_spf', 'Dermaco', '207', '1200', 'India']
--------------------------------------------------------------------------------
ID       Product         brand           qty     price   country
--------------------------------------------------------------------------------
1       Glycolicserum   Ordinary        16      500     USA
2       Night_cream     Cetaphil        52      280     Switzerland
3       Sunscreen       BeautyofJoseon  128     700     Korea
4       Eyecream        Centella        100     400     China
5       Moisturizer     Mamaearth       344     500     India
6       FaceCleanser    HadaLabo        700     900     Japan
7       lipbalm_spf     Dermaco 207     1200    India
--------------------------------------------------------------------------------
 1. Purchase
 2. Restock a product
 3. Exit the application
Enter the option to continue: f
Invalid input. Please enter a number.
Enter the option to continue:
```

Figure 8. test 1 invalid input

4.2. Test 2

Objectives	Test validation of invalid inputs during product purchase and restock
Action	Enter a negative quantity and a non existent value as inputs.
Expected Result	Program should display an error message and ask for valid input
Actual Result	Program displayed “invalid input”
Conclusion	The test was successful.

Table 2. test 2

```
-------------------------------------------------------------------------------
ID       Product         brand           qty     price   country
-------------------------------------------------------------------------------
1       Glycolicserum   Ordinary        16      500     USA
2       Night_cream     Cetaphil        52      280     Switzerland
3       Sunscreen       BeautyofJoseon  128     700     Korea
4       Eyecream        Centella        100     400     China
5       Moisturizer     Mamaearth       344     500     India
6       FaceCleanser    HadaLabo        700     900     Japan
7       lipbalm_spf     Dermaco 207     1200    India
-------------------------------------------------------------------------------
 1. Purchase
 2. Restock a product
 3. Exit the application
Enter the option to continue: 1


Available Products:
1. Glycolicserum (Quantity: 16, Cost Price: 500)
2. Night_cream (Quantity: 52, Cost Price: 280)
3. Sunscreen (Quantity: 128, Cost Price: 700)
4. Eyecream (Quantity: 100, Cost Price: 400)
5. Moisturizer (Quantity: 344, Cost Price: 500)
6. FaceCleanser (Quantity: 700, Cost Price: 900)
7. lipbalm_spf (Quantity: 207, Cost Price: 1200)
Enter your details to generate the bill:
Please enter the name of the customer: kesang
Please enter the phone number of the customer: 9841277311
Enter the ID of the product: -1
Product not available. Please try again
Enter the ID of the product: 9
Product not available. Please try again
Enter the ID of the product:
```

Figure 9 negative quantity and non existent value

4.3. Test 3

Objectives	Test the full purchase process and restocking multiple products
Action	Select option 2, restock more than one item and complete the rest of the process
Expected Result	Program should allow multiple restocks, prints correct the bill in shell and save in txt file
Actual Result	Program allows multiple restocks, prints bill in shell and saves bill in txt file
Conclusion	The test was successful.

Table 3.test 3

```
1 : ['Glycolicserum', 'Ordinary', '136', '800', 'USA']
2 : ['Night_cream', 'Cetaphil', '51', '600', 'Switzerland']
3 : ['Sunscreen', 'BeautyofJoseon', '128', '700', 'Korea']
4 : ['Eyecream', 'Centella', '142', '400', 'China']
5 : ['Moisturizer', 'Mamaearth', '318', '500', 'India']
6 : ['FaceCleanser', 'HadaLabo', '700', '900', 'Japan']
7 : ['lipbalm_spf', 'Dermaco', '207', '1200', 'India']
--------------------------------------------------------------------------------
ID       Product         brand           qty     price   country
--------------------------------------------------------------------------------
1       Glycolicserum   Ordinary        136     800     USA
2       Night_cream     Cetaphil        51      600     Switzerland
3       Sunscreen       BeautyofJoseon  128     700     Korea
4       Eyecream        Centella        142     400     China
5       Moisturizer     Mamaearth       318     500     India
6       FaceCleanser    HadaLabo        700     900     Japan
7       lipbalm_spf     Dermaco 207     1200    India
--------------------------------------------------------------------------------
 1. Purchase
 2. Restock a product
 3. Exit the application
Enter the option to continue: 2
```

```
Enter the ID of the product: 2
Enter quantity to add: 30
Enter new price:
Product restocked successfully!
Do you want to restock another product?(Y/N):y

Available Products:
1. Glycolicserum (Quantity: 336, Cost Price: 1100)
2. Night_cream (Quantity: 79, Cost Price: 600)
3. Sunscreen (Quantity: 128, Cost Price: 700)
4. Eyecream (Quantity: 142, Cost Price: 400)
5. Moisturizer (Quantity: 338, Cost Price: 600)
6. FaceCleanser (Quantity: 667, Cost Price: 1000)
7. lipbalm_spf (Quantity: 207, Cost Price: 1200)
Enter the ID of the product: 2
Enter quantity to add: 40
Enter new price:
Product restocked successfully!
Do you want to restock another product?(Y/N):n

================================================================================
====================
                        We Care - Restock Invoice
                Boudha, Kathmandu | Phone No: 9841277311
--------------------------------------------------------------------------------
Admin Restock Summary:
--------------------------------------------------------------------------------
Product: Night_cream
Quantity Added: 30
New Total Quantity: 79
Updated Price: $ 600
--------------------------------------------------------------------------------
Product: Night_cream
Quantity Added: 40
New Total Quantity: 119
Updated Price: $ 600
--------------------------------------------------------------------------------
```

```
------------------------------------------------------------------------

1 : ['Glycolicserum', 'Ordinary', '336', '1100', 'USA']
2 : ['Night_cream', 'Cetaphil', '119', '600', 'Switzerland']
3 : ['Sunscreen', 'BeautyofJoseon', '128', '700', 'Korea']
4 : ['Eyecream', 'Centella', '142', '400', 'China']
5 : ['Moisturizer', 'Mamaearth', '338', '600', 'India']
6 : ['FaceCleanser', 'HadaLabo', '667', '1000', 'Japan']
7 : ['lipbalm_spf', 'Dermaco', '207', '1200', 'India']
--------------------------------------------------------------------------------
ID       Product         brand           qty     price   country
--------------------------------------------------------------------------------
1       Glycolicserum   Ordinary        336     1100    USA
2       Night_cream     Cetaphil        119     600     Switzerland
3       Sunscreen       BeautyofJoseon  128     700     Korea
4       Eyecream        Centella        142     400     China
5       Moisturizer     Mamaearth       338     600     India
6       FaceCleanser    HadaLabo        667     1000    Japan
7       lipbalm_spf     Dermaco 207     1200    India
--------------------------------------------------------------------------------
```

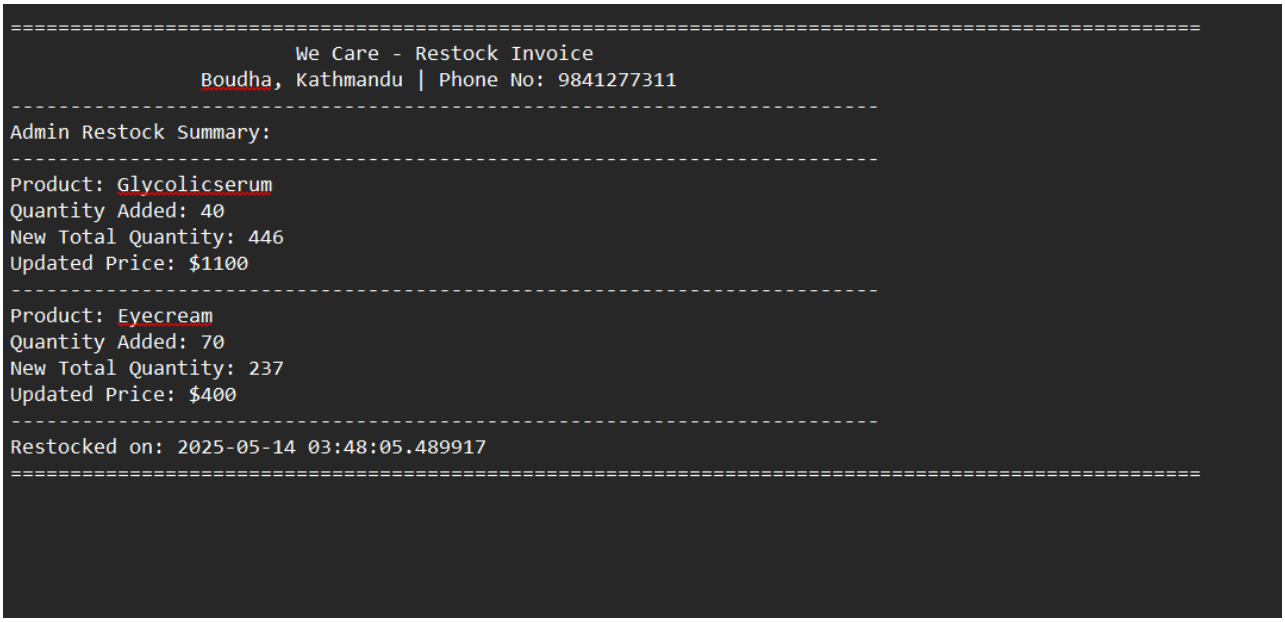


Figure 10. restocking multiple products

4.4. Test 4

Objectives	Test the full sale process and check the output
Action	Select option 1, purchase more than one item and complete the rest of the process
Expected Result	Program should allow multiple purchases, prints correct the bill in shell and save in txt file
Actual Result	Program allows multiple purchases, prints bill in shell and saves bill in txt file
Conclusion	The test was successful.

Table 4.test 4

```
1 : ['Glycolicserum', 'Ordinary', '36', '500', 'USA']
2 : ['Night_cream', 'Cetaphil', '52', '280', 'Switzerland']
3 : ['Sunscreen', 'BeautyofJoseon', '128', '700', 'Korea']
4 : ['Eyecream', 'Centella', '142', '400', 'China']
5 : ['Moisturizer', 'Mamaearth', '344', '500', 'India']
6 : ['FaceCleanser', 'HadaLabo', '700', '900', 'Japan']
7 : ['lipbalm_spf', 'Dermaco', '207', '1200', 'India']
--------------------------------------------------------------------------------
ID       Product         brand           qty     price   country
--------------------------------------------------------------------------------
1       Glycolicserum   Ordinary        36      500     USA
2       Night_cream     Cetaphil        52      280     Switzerland
3       Sunscreen       BeautyofJoseon  128     700     Korea
4       Eyecream        Centella        142     400     China
5       Moisturizer     Mamaearth       344     500     India
6       FaceCleanser    HadaLabo        700     900     Japan
7       lipbalm_spf     Dermaco 207     1200    India
--------------------------------------------------------------------------------
 1. Purchase
 2. Restock a product
 3. Exit the application
Enter the option to continue: 1


Available Products:
1. Glycolicserum (Quantity: 36, Cost Price: 500)
2. Night_cream (Quantity: 52, Cost Price: 280)
3. Sunscreen (Quantity: 128, Cost Price: 700)
4. Eyecream (Quantity: 142, Cost Price: 400)
5. Moisturizer (Quantity: 344, Cost Price: 500)
6. FaceCleanser (Quantity: 700, Cost Price: 900)
7. lipbalm_spf (Quantity: 207, Cost Price: 1200)
Enter your details to generate the bill:
Please enter the name of the customer: kesang
Please enter the phone number of the customer: 9862833422
Enter the ID of the product: 5
please provide the quantity you want to buy:20

Dear kesang you received 6 items for free as part of the offer.
```

```
Dear kesang you received 6 items for free as part of the offer.
Do you want to buy another item? (Y/N): y
Enter the ID of the product: 2
please provide the quantity you want to buy:5

Dear kesang you received 1 items for free as part of the offer.
Do you want to buy another item? (Y/N): n
Do you want your products to be shipped? (Y/N): y
                        We Care

                Boudha, Kathmandu | Phone No: 9841277311

------------------------------------------------------------------------
Customer Info:
------------------------------------------------------------------------
Name of the Customer: kesang
Contact number: 9862833422
Date and time of purchase: 2025-05-14 01:17:35.952020
------------------------------------------------------------------------

Purchase Detail:
------------------------------------------------------------------------------------------------------------------------
Item Name               Quantity        Free    Unit Price      Total
------------------------------------------------------------------------------------------------------------------------
Moisturizer             20              6       1000            $20000
Night_cream             5               1       560             $2800
------------------------------------------------------------------------------------------------------------------------
Shipping Cost: $500
Grand Total: $23300
```

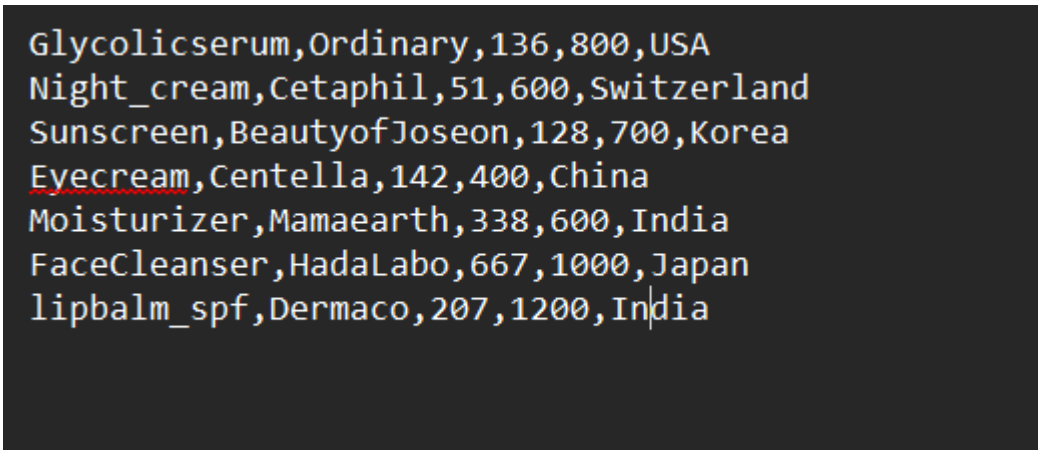
```
=========================================================================================================
                        We Care
                Boudha, Kathmandu | Phone No: 9841277311
-------------------------------------------------------------------------
Customer Details:
-------------------------------------------------------------------------
Name of the Customer: kesang
Contact number: 9862833422
Date and time of purchase: 2025-05-14 01:17:35.952020
-------------------------------------------------------------------------

Purchase Detail:
---------------------------------------------------------------------------------------------------------------------
Item Name        Quantity        Free    Unit Price      Total
---------------------------------------------------------------------------------------------------------------------
Moisturizer     20              6               1000            $20000
Night_cream     5               1               560             $2800
---------------------------------------------------------------------------------------------------------------------
Shipping Cost: $500
Grand Total: $23300
```

Figure 11Purchasing multiple products

4.5. Test 5

Objectives	Test whether the stocks updates or not during restock and sale
Action	Sell a product and deduct quantity and restock the product by adding quantity
Expected Result	Quantity should increase when restocked and decrease when sold in txt file
Actual Result	Quantity is correctly updated in memory and shown in txt file
Conclusion	The test was successful.

Table 5.test 5



```
Enter the option to continue: 1



Available Products:
1. Glycolicserum (Quantity: 136, Cost Price: 800)
2. Night_cream (Quantity: 51, Cost Price: 600)
3. Sunscreen (Quantity: 128, Cost Price: 700)
4. Eyecream (Quantity: 142, Cost Price: 400)
5. Moisturizer (Quantity: 338, Cost Price: 600)
6. FaceCleanser (Quantity: 667, Cost Price: 1000)
7. lipbalm_spf (Quantity: 207, Cost Price: 1200)
Enter your details to generate the bill:
Please enter the name of the customer: kesang
Please enter the phone number of the customer: 9872633822
Enter the ID of the product: 2
please provide the quantity you want to buy:2

Dear kesang you received 0 items for free as part of the offer.
Do you want to buy another item? (Y/N): n
Do you want your products to be shipped? (Y/N): n
```

```
Glycolicserum,Ordinary,136,800,USA
Night_cream,Cetaphil,49,600,Switzerland
Sunscreen,BeautyofJoseon,128,700,Korea
Eyecream,Centella,142,400,China
Moisturizer,Mamaearth,338,600,India
FaceCleanser,HadaLabo,667,1000,Japan
lipbalm_spf,Dermaco,207,1200,India
```

Figure 12. product decreasing in text file

```
Glycolicserum,Ordinary,236,900,USA
Night_cream,Cetaphil,49,600,Switzerland
Sunscreen,BeautyofJoseon,128,700,Korea
Eyecream,Centella,142,400,China
Moisturizer,Mamaearth,338,600,India
FaceCleanser,HadaLabo,667,1000,Japan
lipbalm_spf,Dermaco,207,1200,India
```

```
 1. Purchase
 2. Restock a product
 3. Exit the application
Enter the option to continue: 2


Restocking the products

Available Products:
1. Glycolicserum (Quantity: 236, Cost Price: 900)
2. Night_cream (Quantity: 49, Cost Price: 600)
3. Sunscreen (Quantity: 128, Cost Price: 700)
4. Eyecream (Quantity: 142, Cost Price: 400)
5. Moisturizer (Quantity: 338, Cost Price: 600)
6. FaceCleanser (Quantity: 667, Cost Price: 1000)
7. lipbalm_spf (Quantity: 207, Cost Price: 1200)
Enter the ID of the product: 1
Enter quantity to add: 100
Enter new price: 1100
Product restocked successfully!
Do you want to restock another product?(Y/N):n
```

```
Glycolicserum,Ordinary,336,1100,USA
Night_cream,Cetaphil,49,600,Switzerland
Sunscreen,BeautyofJoseon,128,700,Korea
Eyecream,Centella,142,400,China
Moisturizer,Mamaearth,338,600,India
FaceCleanser,HadaLabo,667,1000,Japan
lipbalm_spf,Dermaco,207,1200,India
```

Figure 13. product increasing in text file

Conclusion

Overall, the Product Wholesale System for WeCare addresses the critical needs of a retail setting in a practical and efficient manner. By concentrating on essential activities such as real-time inventory management, correct billing, and promotional logic integration, the system provides a consistent experience for both employees and customers. One of the most important characteristics is the "buy three, get one free" policy, which activates automatically during checkout and ensures that promotions are handled properly and without the need for manual calculation. Furthermore, the ability to refill items and update prices via the system enables store employees to maintain precise inventory levels and respond quickly to supply changes.

The adoption of Python as the programming language enabled the creation of a versatile and readable system, utilizing data structures such as lists and dictionaries for effective data management. Storing product information in a plain text file kept things simple and light, while also allowing for consistent access and updates. The incorporation of capabilities like invoice production for both client purchases and supplier replenishment helps keep orderly transaction records, which further supports operational transparency. Particularly, this project focuses on automation to reduce repetitive manual activities, limit human error, and increase overall retail process efficiency. It shows how even a smallscale company like WeCare can use digital technologies to improve accuracy, speed up operations, and deliver a better experience for both employees and consumers. This system establishes the groundwork for future enhancements, such as adding a user interface or integrating a database, and shows how technology can alter everyday retail operations.

Appendix

1. Main.py

```
from datetime import datetime
from read import load_data
from write import generate_invoice
from write import restock_invoice
from operations import *

main_loop = True

while main_loop:
    product_data = load_data()

    print()
    for key, value in product_data.items():
        print(key, ":", value)

    print("-" * 80)
    print("ID \t Product \t brand \t\t qty \t price \t country")
    print("-" * 80)
    for key, value in product_data.items():
        print(str(key) + "\t" + value[0] + "\t" + value[1] + "\t" + value[2] + "\t" + value[3] +
"\t" + value[4])

    print("-" * 80)

    print(" 1. Purchase")
    print(" 2. Restock a product")
    print(" 3. Exit the application")

    choice = get_menu_option()
```

```
        print("\n")

    if choice == 1:
        show_products(product_data)

        print("Enter your details to generate the bill:")
        name = input("Please enter the name of the customer: ")
        phone_number = input("Please enter the phone number of the customer: ")

        purchased_items = []
        total = 0
        shipping_cost = 0
        grand_total = 0

        while True:
            product_id = get_product_id(product_data)
            qty_available = int(product_data[product_id][2])
            product_quantity, free_qty = get_purchase_qty(qty_available)

            print("\nDear", name, "you received", free_qty, "items for free as part of the offer.")

            product_data[product_id][2] = str(qty_available - product_quantity - free_qty)

            product_name = product_data[product_id][0]
            unit_price = int(product_data[product_id][3]) * 2
            total_price = unit_price * product_quantity

            purchased_items.append([product_name, product_quantity, unit_price, total_price, free_qty])
            total += total_price
            more = input("Do you want to buy another item? (Y/N): ").lower()
```

```
            if more not in ['y', 'yes']:
                break

        shipping_choice = input("Do you want your products to be shipped? (Y/N): ").lower()
        if shipping_choice == "y":
            shipping_cost = 500

        grand_total = total + shipping_cost
        today_date_and_time = datetime.now()

        save_data(product_data)

        print("\t\t\tWe Care \n")
        print("\t\tBoudha, Kathmandu | Phone No: 9841277311\n")
        print("-------------------------------------------------------------------------")
        print("Customer Info:")
        print("-------------------------------------------------------------------------")
        print("Name of the Customer:", name)
        print("Contact number:", phone_number)
        print("Date and time of purchase:", str(today_date_and_time))
        print("-------------------------------------------------------------------------")
        print("\nPurchase Detail:")
        print("-------------------------------------------------------------------------------------------------")
        print("Item Name \t\t Quantity \t Free \t Unit Price \t Total")
        print("-------------------------------------------------------------------------------------------------")
        for i in purchased_items:
            print(i[0] + "\t\t" + str(i[1]) + "\t\t" + str(i[4]) + "\t" + str(i[2]) + "\t\t$" + str(i[3]))
        print("-------------------------------------------------------------------------------------------------")
```

```
        if shipping_cost > 0:
            print("Shipping Cost: $" + str(shipping_cost))
        print("Grand Total: $" + str(grand_total))
        print("\n")

        filename = "invoice_" + str(today_date_and_time).replace(":", "-").replace(" ", "_") + ".txt"
        generate_invoice(filename, name, phone_number, today_date_and_time, purchased_items, shipping_cost, grand_total)

    elif choice == 2:
        print("Restocking the products")
        restock_records = []
        while True:
            show_products(product_data)
            restock_id = get_product_id(product_data)
            qty = get_restock_qty()
            cost = input("Enter new price: ")
            product_data[restock_id][2] = str(int(product_data[restock_id][2]) + qty)
            if cost:
                product_data[restock_id][3] = cost
            save_data(product_data)
            print("Product restocked successfully!")
            restock_records.append([product_data[restock_id][0], qty, product_data[restock_id][2], product_data[restock_id][3]])

            more=input("Do you want to restock another product?(Y/N):").lower()
            if more=="y":
                continue
            else:
                break
```

```
            today_date_and_time = datetime.now()
            filename = "restock_invoice_" + str(today_date_and_time).replace(":", "-
        ").replace(" ", "_") + ".txt"
            restock_invoice(filename, restock_records, today_date_and_time)

            print("\n" + "="*100)
            print("\t\t\tWe Care - Restock Invoice")
            print("\t\tBoudha, Kathmandu | Phone No: 9841277311")
            print("-------------------------------------------------------------------------")
            print("Admin Restock Summary:")
            print("-------------------------------------------------------------------------")
            for record in restock_records:
                print("Product:", record[0])
                print("Quantity Added:", record[1])
                print("New Total Quantity:", record[2])
                print("Updated Price: $", record[3])
                print("-------------------------------------------------------------------------")


        elif choice == 3:
            main_loop = False
            print("Thank you for using the system\n")
```

2. read.py

```
        def load_data():
            product_data = {}
            file = open("data.txt", "r")
            ff = file.readlines()
            item_id = 1
            for data in ff:
                data = data.replace("\n", "").split(",")
```

```
            product_data[item_id] = data
            item_id += 1
        file.close()
        return product_data
```

3. write.py

```
def generate_invoice(file_path, name, phone_number, datetime_now, items, shipping_cost, grand_total):
    file = open(file_path, "a")
    file.write("\n" + "="*100 + "\n")
    file.write("\t\t\tWe Care\n")
    file.write("\t\tBoudha, Kathmandu | Phone No: 9841277311\n")
    file.write("-------------------------------------------------------------------------\n")
    file.write("Customer Details:\n")
    file.write("-------------------------------------------------------------------------\n")
    file.write("Name of the Customer: " + name + "\n")
    file.write("Contact number: " + phone_number + "\n")
    file.write("Date and time of purchase: " + str(datetime_now) + "\n")
    file.write("-------------------------------------------------------------------------\n")
    file.write("\nPurchase Detail:\n")
    file.write("-------------------------------------------------------------------------------------------------\n")
    file.write("Item Name \t Quantity \t Free \t Unit Price \t Total\n")
    file.write("-------------------------------------------------------------------------------------------------\n")
    for item in items:
        file.write(item[0] + "\t" + str(item[1]) + "\t\t" + str(item[4]) + "\t\t" + str(item[2]) + "\t\t$" + str(item[3]) + "\n")
    file.write("-------------------------------------------------------------------------------------------------\n")
    if shipping_cost > 0:
        file.write("Shipping Cost: $" + str(shipping_cost) + "\n")
    file.write("Grand Total: $" + str(grand_total) + "\n")
```

```
        file.close()
    def restock_invoice(file_path, restock_records, datetime_now):
        with open(file_path, "a") as file:
            file.write("\n" + "="*100 + "\n")
            file.write("\t\t\tWe Care - Restock Invoice\n")
            file.write("\t\tBoudha, Kathmandu | Phone No: 9841277311\n")
            file.write("-------------------------------------------------------------------------\n")
            file.write("Admin Restock Summary:\n")
            file.write("-------------------------------------------------------------------------\n")
            for record in restock_records:
                file.write("Product: " + record[0] + "\n")
                file.write("Quantity Added: " + str(record[1]) + "\n")
                file.write("New Total Quantity: " + str(record[2]) + "\n")
                file.write("Updated Price: $" + str(record[3]) + "\n")
                file.write("-------------------------------------------------------------------------\n")
            file.write("Restocked on: " + str(datetime_now) + "\n")
            file.write("="*100 + "\n")
```

4. operation.py

```
    def show_products(product_data):
        print("\nAvailable Products:")
        for key, value in product_data.items():
```

```
        print(str(key) + ". " + value[0] + " (Quantity: " + value[2] + ", Cost Price: " +
value[3] + ")")

def get_menu():
    while True:
        try:
            choice = int(input("Enter the option to continue: "))
            if choice == 1 or choice == 2 or choice == 3:
                return choice
            else:
                print("Please choose 1, 2, or 3.")
        except:
            print("Invalid input. Please enter a number.")

def get_product_id(product_data):
    while True:
        try:
            product_id = int(input("Enter the ID of the product: "))
            if product_id >= 1 and product_id <= len(product_data):
                return product_id
            else:
                print("Product not available. Please try again")
        except:
            print("Please enter a valid ID")

def get_purchase_qty(available_qty):
    while True:
        try:
            quantity = int(input("please provide the quantity you want to buy:"))
            free = quantity // 3
            total_needed = quantity + free
            if quantity > 0 and total_needed <= available_qty:
```

```
                return quantity, free
            else:
                print("The quantity is not available")
        except:
            print("Invalid quantity")

def get_restock_qty():
    while True:
        try:
            qty = int(input("Enter quantity to add: "))
            if qty > 0:
                return qty
            else:
                print("Quantity must be greater than 0.")
        except:
            print("Invalid input. Enter a valid number.")

def save_data(product_data):
    file = open("data.txt", "w")
    for values in product_data.values():
        file.write(",".join(values) + "\n")
    file.close()
```

References

AS, R. (2024, july 25). *Set in Python: Everything You Need to Know About It*. Retrieved from simplilearn: https://www.simplilearn.com/tutorials/python-tutorial/set-in-python

draw.io. (2024, 7 18). Retrieved from Computerhope: https://www.computerhope.com/jargon/d/drawio.htm

Jaishree. (2025, may 10). *Mastering Python IDLE: A Beginner's Guide*. Retrieved from guvi.in: https://www.guvi.in/hub/python/what-is-idle/

w3schools. (n.d.). *Dictionaries in Python*. Retrieved from w3schools: https://www.geeksforgeeks.org/python-dictionary/

w3schools. (n.d.). *Python Lists*. Retrieved from w3schools: https://www.w3schools.com/python/python_lists.asp