CS4051NI Fundamentals of Computing

70% Individual Coursework

Final Submission

2024/25 Spring

Student Name: Anamika Bhattarai

London Met ID: 24046705

ID: NP01CP4A240280

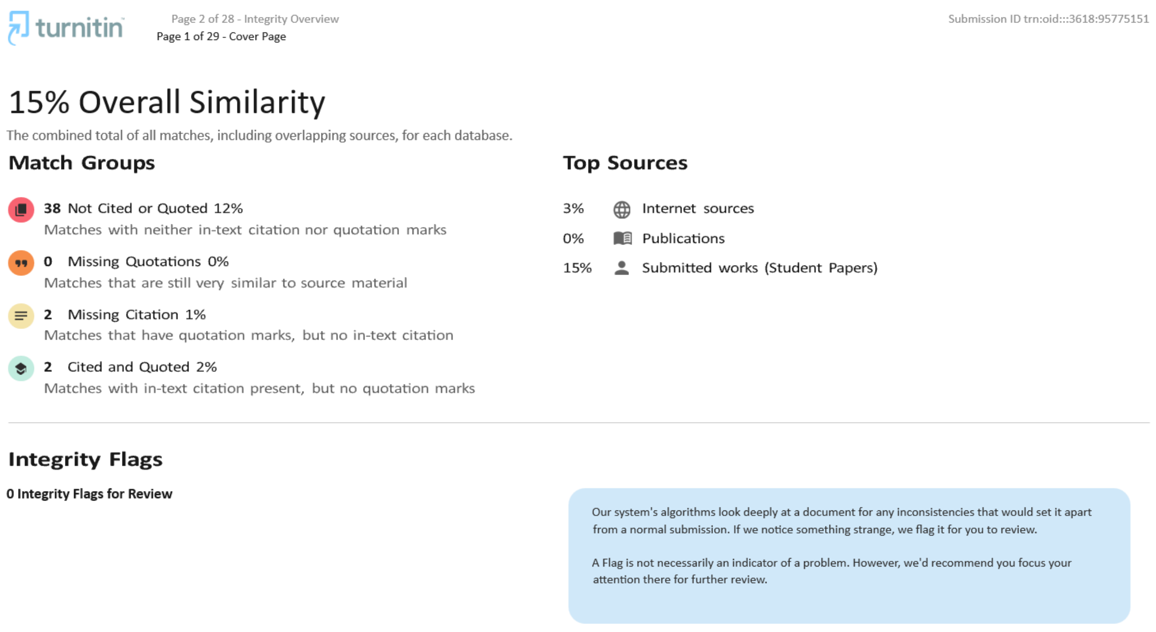
Assignment Due Date: 14th May,2025

Assignment Submission Date: 14th May,2025

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.

Table of Contents

Similarity Report



turnitin Page 2 of 28 - Integrity Overview

Page 1 of 29 - Cover Page

Submission ID trn:oid:::3618:95775151

15% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **38** Not Cited or Quoted 12%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **2** Missing Citation 1%
  Matches that have quotation marks, but no in-text citation
- **2** Cited and Quoted 2%
  Matches with in-text citation present, but no quotation marks

Top Sources

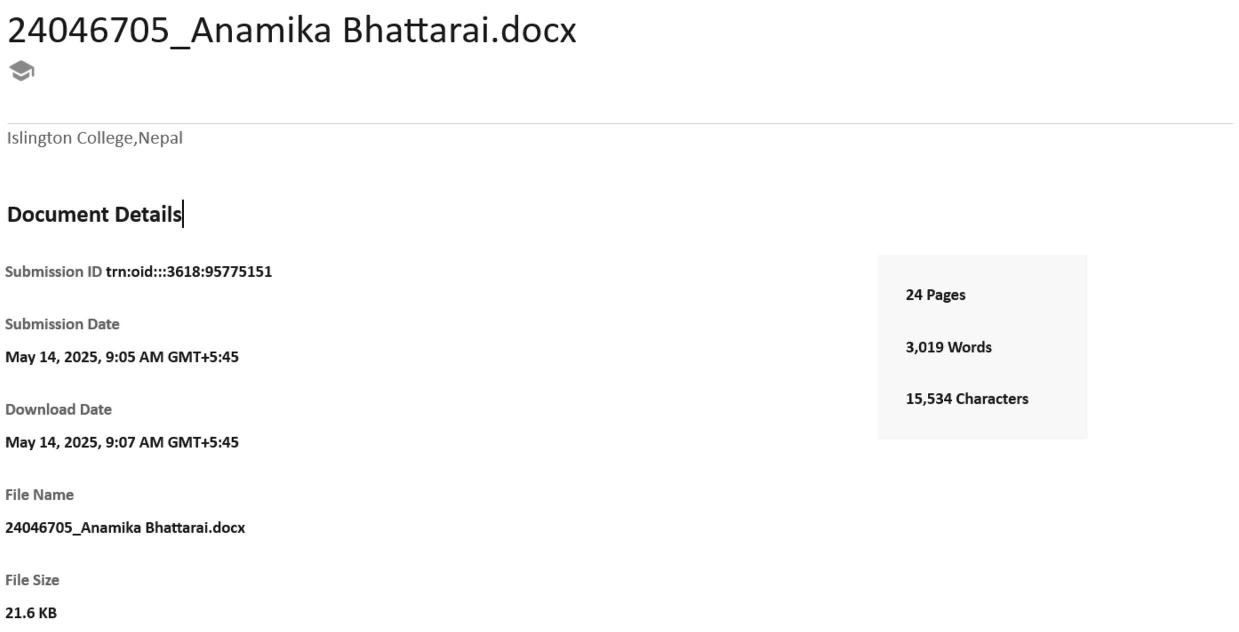
3%	Internet sources
0%	Publications
15%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.



24046705_Anamika Bhattarai.docx

Islington College,Nepal

Document Details

Submission ID trn:oid:::3618:95775151

Submission Date

May 14, 2025, 9:05 AM GMT+5:45

Download Date

May 14, 2025, 9:07 AM GMT+5:45

File Name

24046705_Anamika Bhattarai.docx

File Size

21.6 KB

24 Pages

3,019 Words

15,534 Characters

1.Introduction

This project is designed to develop the skin care sale system named as “WeCare” using the python programming language. WeCare inventory management system has effective operations such as managing stocks, sales offer and tracking the customers transactions. This project provides the core programming applications of python like file handling, modular design, data structures etc. The key point or essential point to be noted for the project is that through this project we get to know the importance of proper data representation and persistent storage in real-world applications. By using dictionaries and lists efficiently, we ensured fast access and scalability of the system.

Aims and Objectives

The main purpose for the project is to design and develop a Python-based inventory management system for a beauty product retail store, integrating a promotional sales policy (buy 3 get 1 free).

Objectives

- To display a list of available products by reading data from the structured text file.
- To create a python-based system that manages product inventory for a beauty and skincare store.
- To ensure product stock is accurately updated after each sale or restock event.
- To keep the system running in loop, allowing multiple operations until the admin decides to exit.
- To automatically generate sales and restock invoices with relevant details for each transaction.
- To implement a promotional offer (buy 3 get 1 free) during the sales process.
- To generate invoices that include detailed transaction data such as product names, quantities (including free items), customer/vendor names, and total cost.

Tools Used:

Python

“Python is high level programming language developed by Guido van Rossum in 1991.” (Python, n.d.).

IDLE

Integrated development and learning environment are the base of this project where we do python code.

Ms Word

We use MS-Word to make the report/documentation for this project. The components in the Ms-Word provides effective use of tables, citation used for the report part.

Notepad

We use notepad to make text file use for this project.

Draw.io

We use draw.io to draw the flowchart of the program and it helps us to make flowchart easily and can easily style with different background.

Data Structures

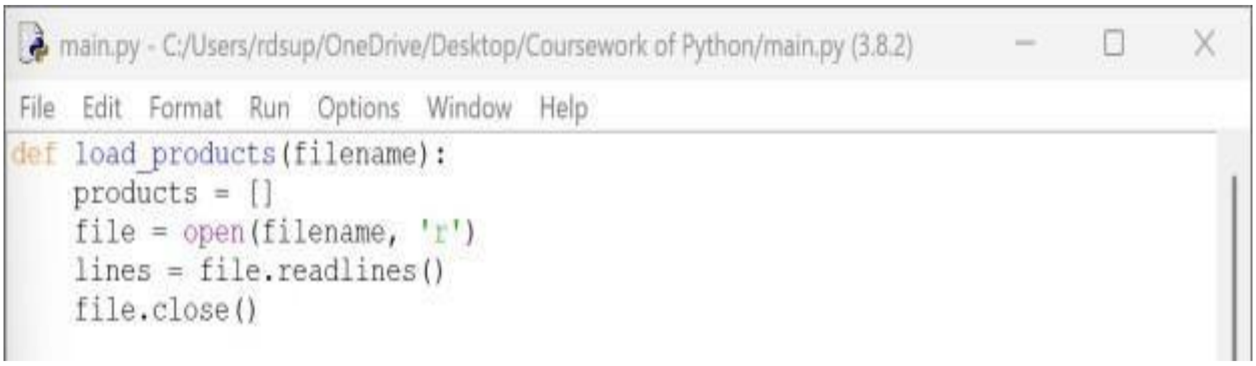
Some of the Data structures (primitive data types /collection data types) used in this project are given below:

String, Integer, Float, Boolean are the primitive data types. String represents characters. int represents integers, float represents real numbers, and Boolean represents Boolean values true or false. Examples: “hello World” is string ,1 is the integer data type, True is Boolean data type and 3.5 is the float data type.

Collection datatypes like list, tuples are mentioned below:

Lists:

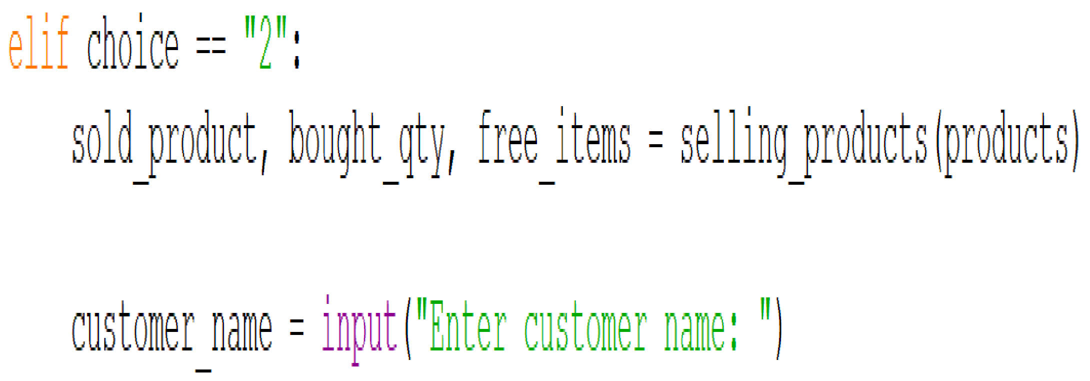
List is the collection of Python objects which is mutable in nature and the elements in list can be added or removed. It is ordered sequence of information, accessible by index. A list can contain element of same data type or as well as different data type. List Indices starts at zero (0). In the image below products is a list


main.py - C:/Users/rdsup/OneDrive/Desktop/Coursework of Python/main.py (3.8.2)
File Edit Format Run Options Window Help

```
def load_products(filename):
    products = []
    file = open(filename, 'r')
    lines = file.readlines()
    file.close()
```

Tuple:

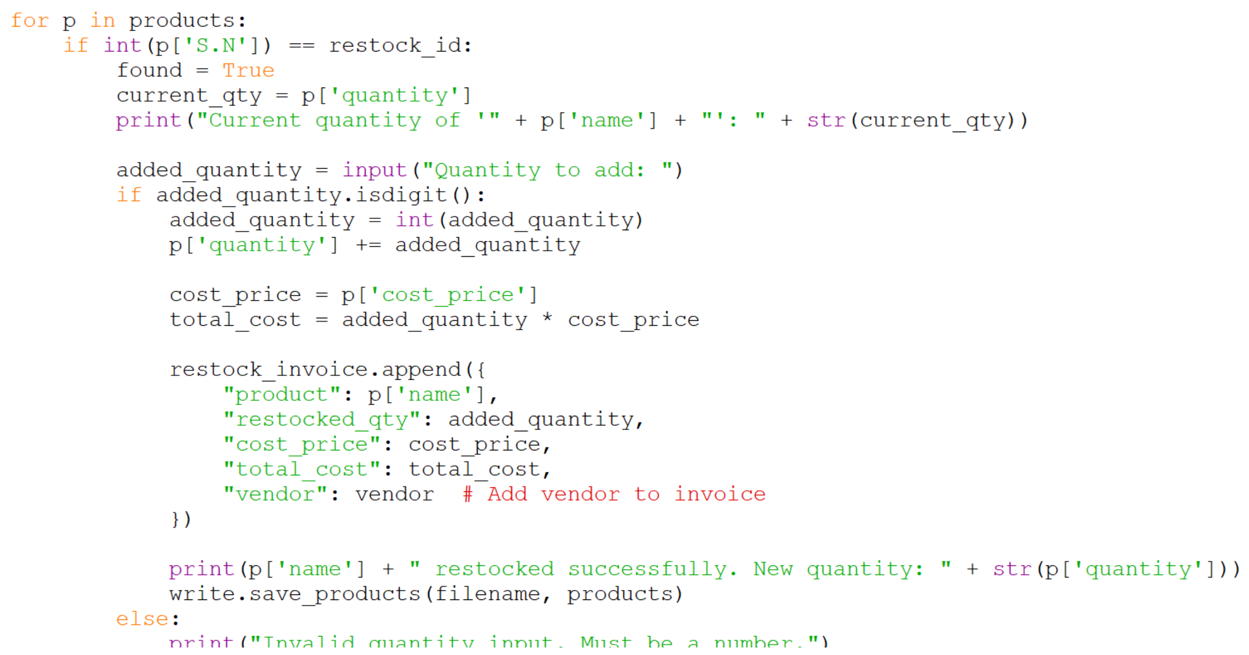
A tuple is a built-in data type in Python that allows you to store a group of values together in a specific sequence. It is immutable.



```
elif choice == "2":
    sold_product, bought_qty, free_items = selling_products(products)

    customer_name = input("Enter customer name: ")
```

Dictionary:

"It is an unordered collection of data values, used to store data values like a map, which, unlike other Data Types that hold only a single value as an element, Dictionary holds the key: value pair. Key value is provided in the dictionary to make more optimized. Indexing of Python Dictionary is done with the help of keys." (geeksforgeeks, 2024)



```python
for p in products:
    if int(p['S.N']) == restock_id:
        found = True
        current_qty = p['quantity']
        print("Current quantity of '" + p['name'] + "': " + str(current_qty))

        added_quantity = input("Quantity to add: ")
        if added_quantity.isdigit():
            added_quantity = int(added_quantity)
            p['quantity'] += added_quantity

            cost_price = p['cost_price']
            total_cost = added_quantity * cost_price

            restock_invoice.append({
                "product": p['name'],
                "restocked_qty": added_quantity,
                "cost_price": cost_price,
                "total_cost": total_cost,
                "vendor": vendor  # Add vendor to invoice
            })

            print(p['name'] + " restocked successfully. New quantity: " + str(p['quantity']))
            write.save_products(filename, products)
        else:
            print("Invalid quantity input. Must be a number.")
```

2.Discussion and Analysis

Algorithm

Step 1: Start the program

Step 2: Ask user to choose the option displayed on the screen

Step 3: if user selects option 1 go to step 2

Step 4: if user selects the option 2 go to step 5

Step 5: Ask user for product ID. If Id is valid then go to step 6, else go to step 5 by showing invalid message.

Step 6: Ask user how many products he/she wants. If input is valid then go to step 7, else go to step 6 by showing invalid message.

Step 7: Ask user do you want to buy more. If yes, then go to step 5 else go to step 8.

Step 8: Ask the user to input a customer name.

Step 9: Display the invoice of purchase products and go to step 2

Step 10: if user selects option 3

Step 11: Ask for the vendor’s name.

Step 12: ask product id for restock.

Step 13: display the information of current quantity of product and ask for the quantity to add.

Step 14: display the message of restock successfully.

Step 15: Ask user do you want to restock another product. If yes then go to step 12 else go to step 16.

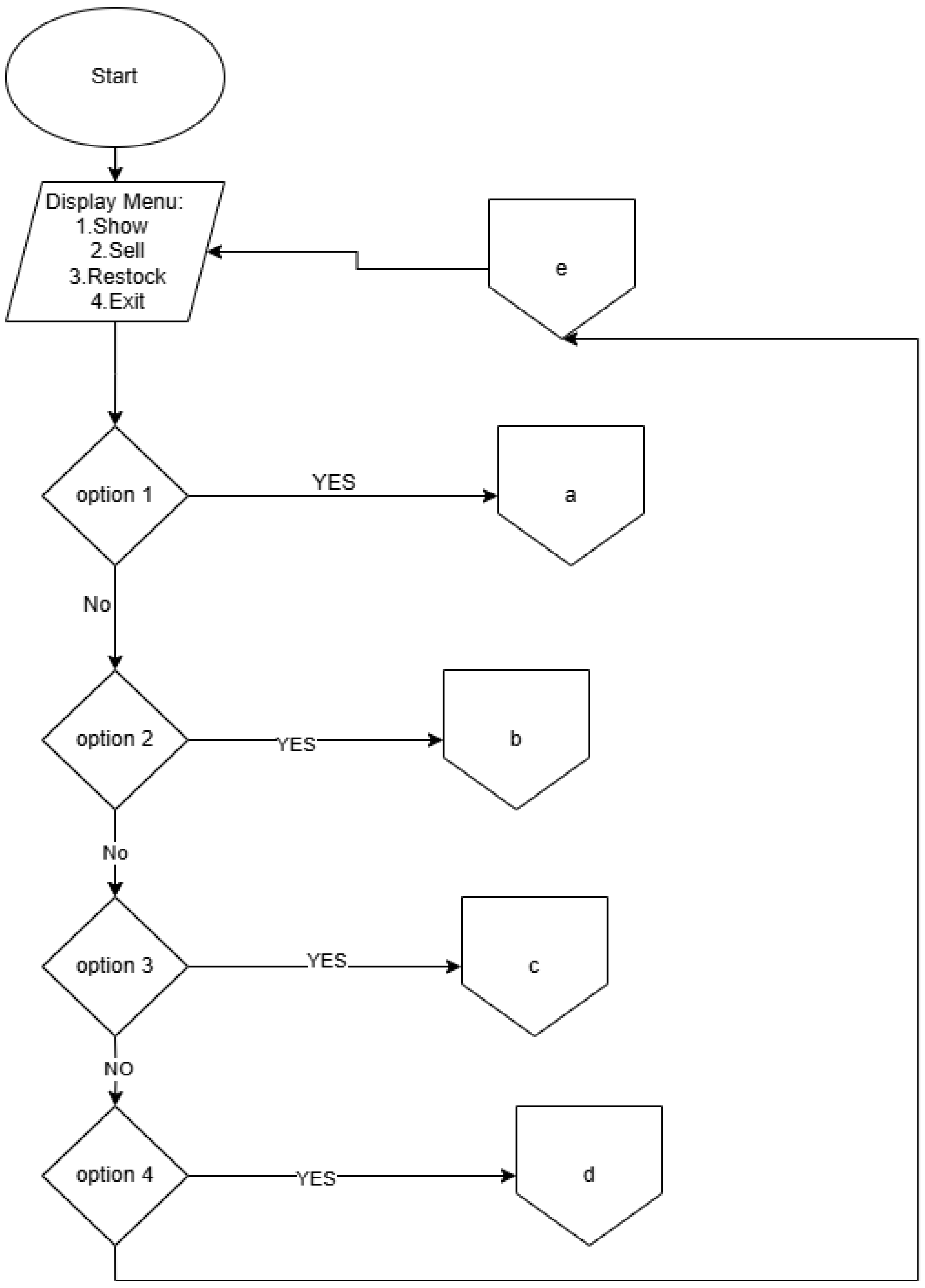
Step 16: Display the restock invoice and go to step 2

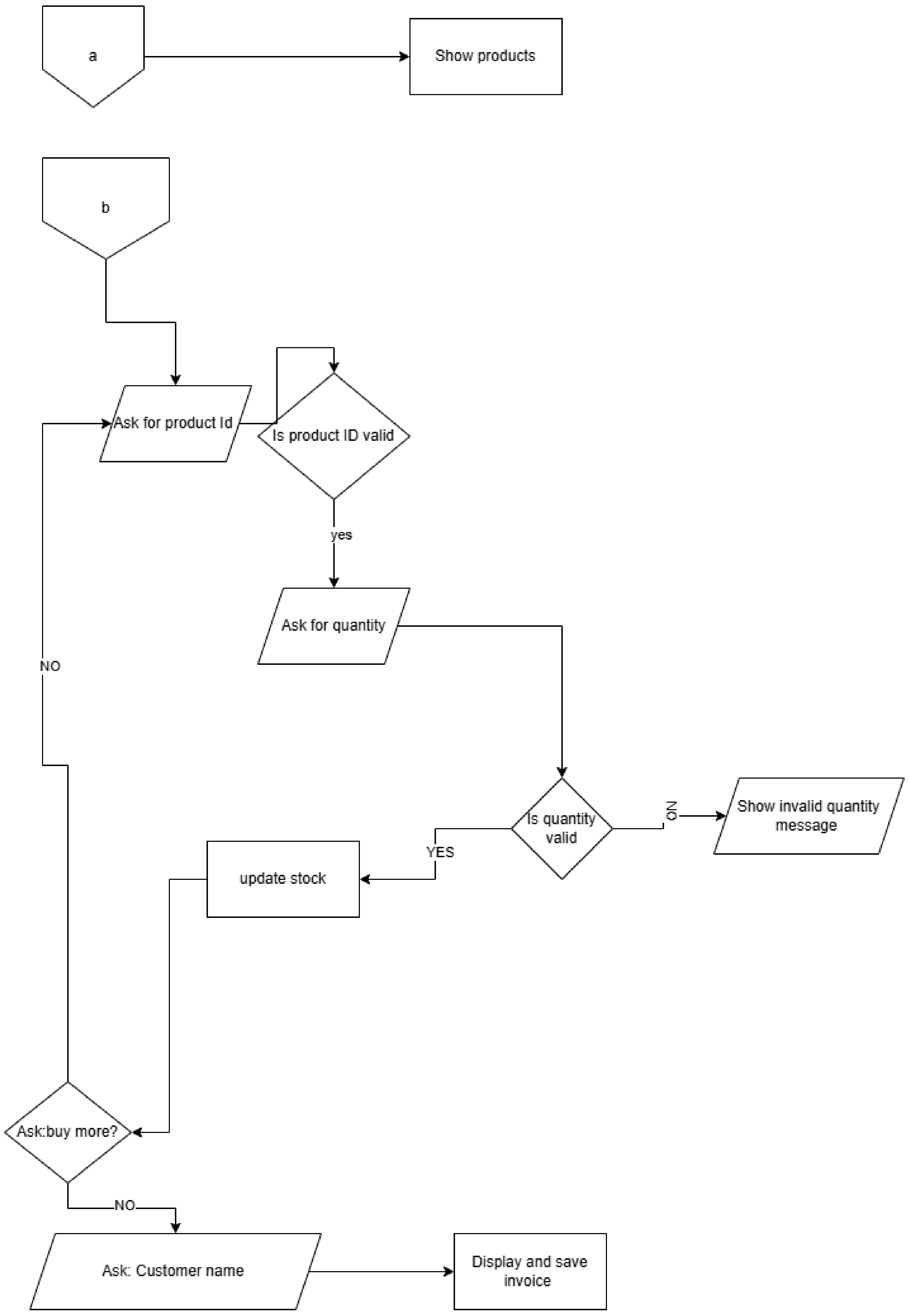
Step 17: if user selects option 4 then closing message is displayed on the screen and go to step 19

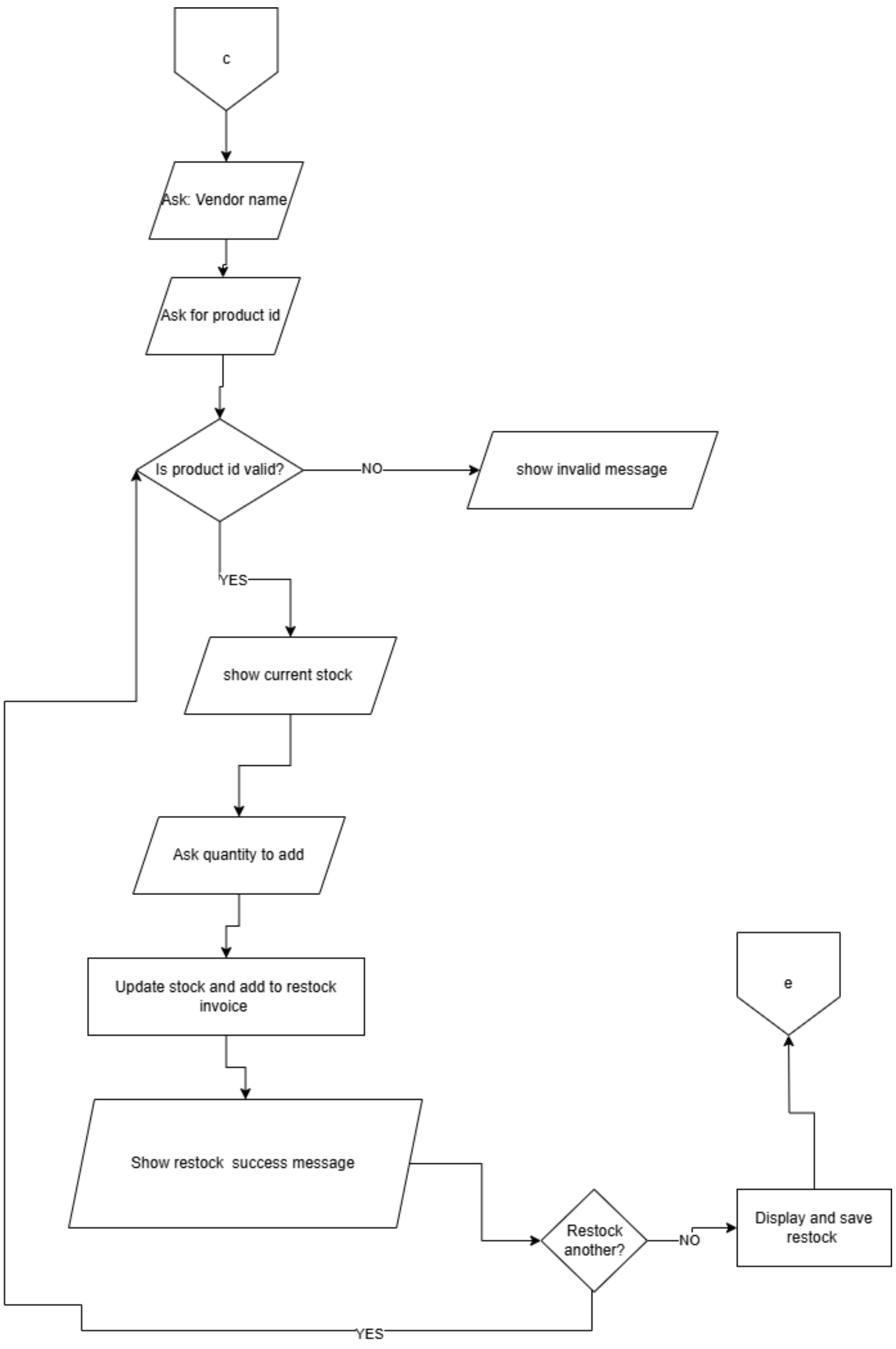
Step 18: if the user selects any other number rather than (1-4) then invalid choice message is displayed and go to step 2.

Step 19: Stop the program.

Flowchart


Start
Display Menu:
1.Show
2.Sell
3.Restock
4.Exit
e
option 1
YES
a
No
option 2
YES
b
No
option 3
YES
c
NO
option 4
YES
d


a
Show products
b
Ask for product Id
Is product ID valid
yes
Ask for quantity
Is quantity valid
NO
Show invalid quantity message
YES
update stock
NO
Ask:buy more?
NO
Ask: Customer name
Display and save invoice


c
Ask: Vendor name
Ask for product id
Is product id valid?
NO
show invalid message
YES
show current stock
Ask quantity to add
Update stock and add to restock invoice
Show restock success message
Restock another?
NO
Display and save restock
e
YES

YES

Pseudocode

Operations file

```
Print "Available Products" header

Print table column headers: S.N, Product, Brand, Stock, C.P, S.P, Origin

Print separator line


For each product p in products:

    Calculate selling price as cost_price * 2


    Format serial number to fixed width (6 spaces)

    Format product name to fixed width (20 spaces)

    Format brand to fixed width (15 spaces)

    Format stock quantity to fixed width (6 spaces)

    Format cost price to fixed width (11 spaces)

    Format selling price to fixed width (11 spaces)

    Get origin


    Print all formatted fields concatenated in one line

Loop forever:

    Prompt user to enter Product ID (S.N)

    If input is a valid digit:

        Set found flag to False

        For each product in products:

            If product's S.N matches user input:

                Set found to True

                Loop forever:
                    Prompt user for quantity to buy
                    If quantity is a valid digit:
                        Calculate total_needed = quantity
                        Calculate free_items = total_needed // 3  (Buy 2 get 1 free offer)
                        Calculate total_with_offer = total_needed + free_items

                        If total_with_offer <= product's stock quantity:
                            Reduce product's quantity by total_with_offer

                            Ask user if they want to buy more (yes/no)
                            If no:
                                Return product, total_needed, free_items
                            Else:
                                Break inner loop to select another product
                        Else:
                            Print "Not enough stock. Try smaller quantity."
                    Else:
                        Print "Enter a valid number."
                Break from product search loop
        If not found:
            Print "Product ID not found."
    Else:
        Print "Enter a valid number."
Initialize empty list restock_invoice
```

```
Prompt user to enter vendor name


Loop forever:
    Prompt user to enter Product ID to restock
    If input is a valid integer:
        Set found flag to False
        For each product p in products:
            If product's S.N matches restock_id:
                Set found to True
                Print current quantity of product


                Prompt user for quantity to add
                If quantity is a valid digit:
                    Convert to integer and add to product's quantity


                    Calculate total_cost = added_quantity * cost_price


                    Append to restock_invoice:
                        product name, restocked_qty, cost_price, total_cost, vendor


                    Print success message with new quantity


                    Call write.save_products(filename, products)
                Else:
                    Print "Invalid quantity input. Must be a number."
            Break from product search loop

        If not found:
            Print "Invalid Product ID. Please try again."
    Else:
        Print "Invalid input. Please enter numeric Product ID."

    Prompt user if want to restock another product (Y/N)
    If user input is not 'Y':
        Break loop

If restock_invoice is not empty:
    Call write.create_restock_invoice(restock_invoice)
    Call write.display_restock_invoice()
Else:
    Print "No products were restocked, so no restock invoice was created."
```

Write file

```
Open file with given filename in write mode

For each product p in products:
    Create a line string by concatenating:
        p['S.N'], p['name'], p['brand'], p['quantity'], p['cost_price'], p['origin']
    Separate each field by commas
    Add newline character at the end

    Write the line string to the file

Close the file
Import datetime module
Get current date and time as formatted string "YYYY-MM-DD HH:MM:SS"

Open file "restock_bill.txt" in write mode

Write header lines:
    "=============== Restock Invoice ===================="
    "Date & Time: " + current date and time string
    "===================================================="

If restock_invoice is not empty:
    Write "Vendor Name: " + vendor name from first item in restock_invoice

Write separator line
"====================================================="

For each item in restock_invoice:
    Write "Product: " + item's product name
    Write "Quantity Restocked: " + item's restocked quantity
    Write "Cost Price per Unit: " + item's cost price
    Write "Total Restock Cost: " + item's total cost
    Write a separator line "--------------------------"

Write footer line "=========================="

Close the file
Print header "==== Restock Invoice ===="
Print separator line "=========================="

Open file "restock_bill.txt" in read mode
Read all contents of the file
Print the contents

Close the file
```

Read file

```
Function read_data(filename):
    Initialize empty list products

    Open file with given filename in read mode
    Read all lines from the file into lines list
    Close the file

    For each line in lines:
        Split line by comma into list index
    If length of index is 6:
        serialno = index[0]
        name = index[1]
        brand = index[2]
        quantity = convert index[3] to integer
        cost_price = convert index[4] to float
        origin = index[5] with newline character removed

        Create a product dictionary with keys:
            'S.N' : serialno
            'name' : name
            'brand' : brand
            'quantity' : quantity
            'cost_price' : cost_price
            'origin' : origin

        Append product dictionary to products list

  Return products list
```

Main file

```
Import required functions from read, write, and operations modules
Import datetime module
```

```
Define function main ():
    Set filename to "products.txt"
    Call read_data(filename) to get products list

    Loop forever:
        Print main menu:
            1. Show Products
            2. Sell Product
            3. Restock Product
            4. Exit

        Prompt user for choice

        If choice is "1":
            Call show_producttable(products)

        Else if choice is "2":
            Call selling_products(products) and get sold_product, bought_qty, free_items

            Prompt user for customer name

            Calculate total_price = bought_qty * sold_product['cost_price'] * 2
            Get current date and time as string

            Initialize invoice_text as empty list
            Append invoice header and details to invoice_text:
                - Date & Time
                - Customer Name
                - Product details (name, brand, origin)
                - Quantity Purchased
                - Free Items
                - Price per unit
                - Total Price

            For each line in invoice_text:
                Print line to screen

            Open "invoice.txt" in write mode
            Write all lines from invoice_text to file
            Close file

            Call save_products(filename, products) to update stock

        Else if choice is "3":
            Call restock_product(filename, products)

        Else if choice is "4":
            Print exit message
            Break loop
        Else:
            Print "Invalid choice" message

    Call main ()
```

3.Program

Implementation of the program

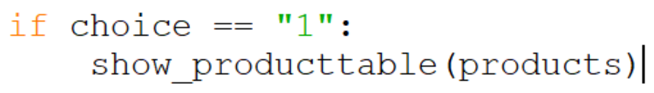
The whole program is differentiated in 4 different python files which are write file, read file, operations file and main file. They perform essential role in every line of the code.

Main.py

It is the starting point of WeCare Skin product management system. It gives option menu to user.

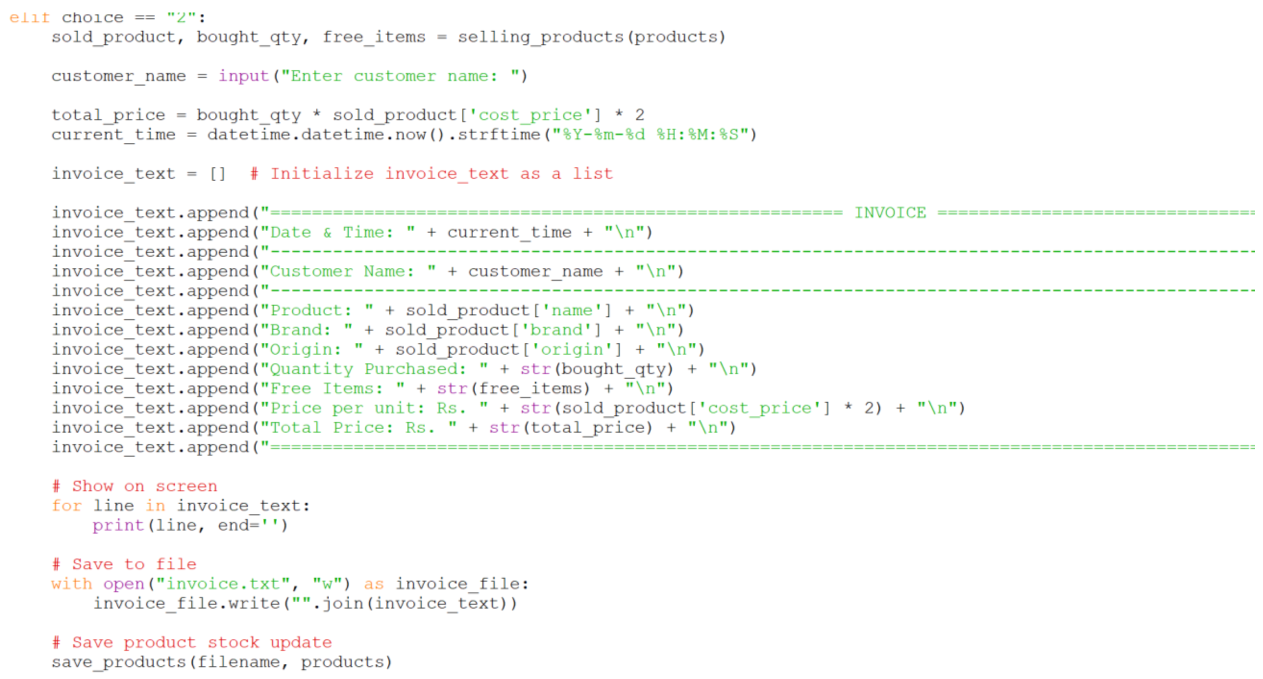
```python
def main():
    """
    ------------------------------------------------------------------------
    Main function to run the WeCare Skin Products Management system.
    ------------------------------------------------------------------------

    Description:
        This function serves as the entry point for the application. It loads
        product data, displays a menu to the user, and handles user choices
        for showing products, selling products, restocking, and exiting the
        program. It also manages invoice creation and product stock updates.

    ------------------------------------------------------------------------
    Parameters:
        None

    ------------------------------------------------------------------------
    Returns:
        None
        This function does not return anything.
    ------------------------------------------------------------------------
    """
    # Function code remains unchanged

    filename = "products.txt"
    products = read_data(filename)

    while True:
        print("\n==== WeCare Skin Products Management ====")
        print("1. Show Products")
        print("2. Sell Product")
        print("3. Restock Product")
        print("4. Exit")
        print("==========================================")

        choice = input("Enter your choice (1-4): ")

        if choice == "1":
            show_producttable(products)

        elif choice == "2":
```

If the user choose option 1: It displays all the available product using a table format. It shows product ID, name, brand, how many items are available in stock, the price and the origin.



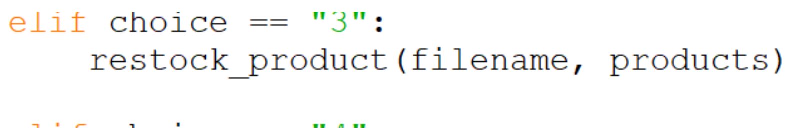
```python
if choice == "1":
    show_producttable(products)
```

If the user choose option 2: The system asks which product the user want to sell and how many the customer want. Apply buy 3 and get 1 free offer. Customer name is taken then total price is calculated. An invoice is created and saved to a text file called invoice.txt. The product stock is updated and saved to product .txt.



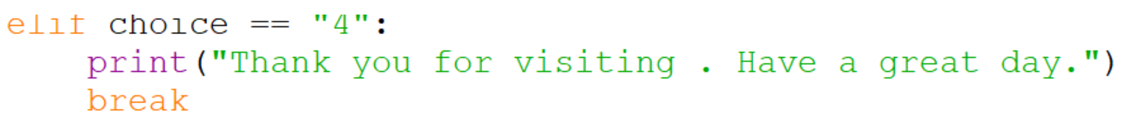
```
elif choice == "2":
    sold_product, bought_qty, free_items = selling_products(products)

    customer_name = input("Enter customer name: ")

    total_price = bought_qty * sold_product['cost_price'] * 2
    current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    invoice_text = []  # Initialize invoice_text as a list

    invoice_text.append("======================================================== INVOICE ===============================
    invoice_text.append("Date & Time: " + current_time + "\n")
    invoice_text.append("-----------------------------------------------------------------------------------------------
    invoice_text.append("Customer Name: " + customer_name + "\n")
    invoice_text.append("-----------------------------------------------------------------------------------------------
    invoice_text.append("Product: " + sold_product['name'] + "\n")
    invoice_text.append("Brand: " + sold_product['brand'] + "\n")
    invoice_text.append("Origin: " + sold_product['origin'] + "\n")
    invoice_text.append("Quantity Purchased: " + str(bought_qty) + "\n")
    invoice_text.append("Free Items: " + str(free_items) + "\n")
    invoice_text.append("Price per unit: Rs. " + str(sold_product['cost_price'] * 2) + "\n")
    invoice_text.append("Total Price: Rs. " + str(total_price) + "\n")
    invoice_text.append("===============================================================================================

    # Show on screen
    for line in invoice_text:
        print(line, end='')

    # Save to file
    with open("invoice.txt", "w") as invoice_file:
        invoice_file.write("".join(invoice_text))

    # Save product stock update
    save_products(filename, products)
```

If the user choose option 3: The system asks which product is being restocked. The number of new items is added. The stock is updated. A restock invoice is generated and saved in a file .This helps to keep records of where and how many items were restored.



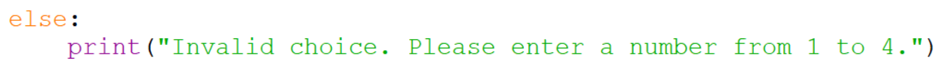
```
elif choice == "3":
    restock_product(filename, products)
```

If the user choose option 4: Programs stops and show the message.



```
elif choice == "4":
    print("Thank you for visiting . Have a great day.")
    break
```

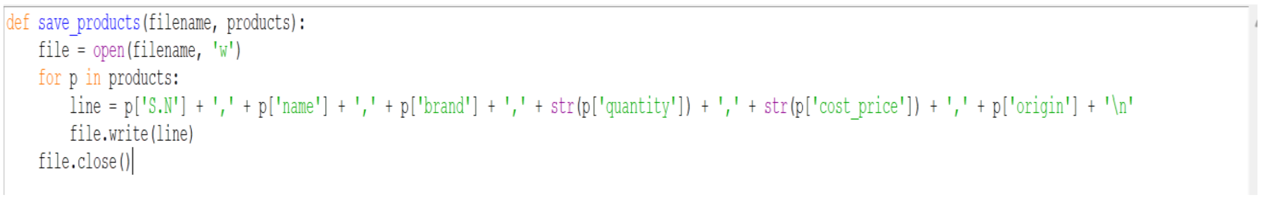
Wrong input handling is done when the user enters input other than (1-4) and show invalid choice messages.



```
else:
    print("Invalid choice. Please enter a number from 1 to 4.")
```

write.py

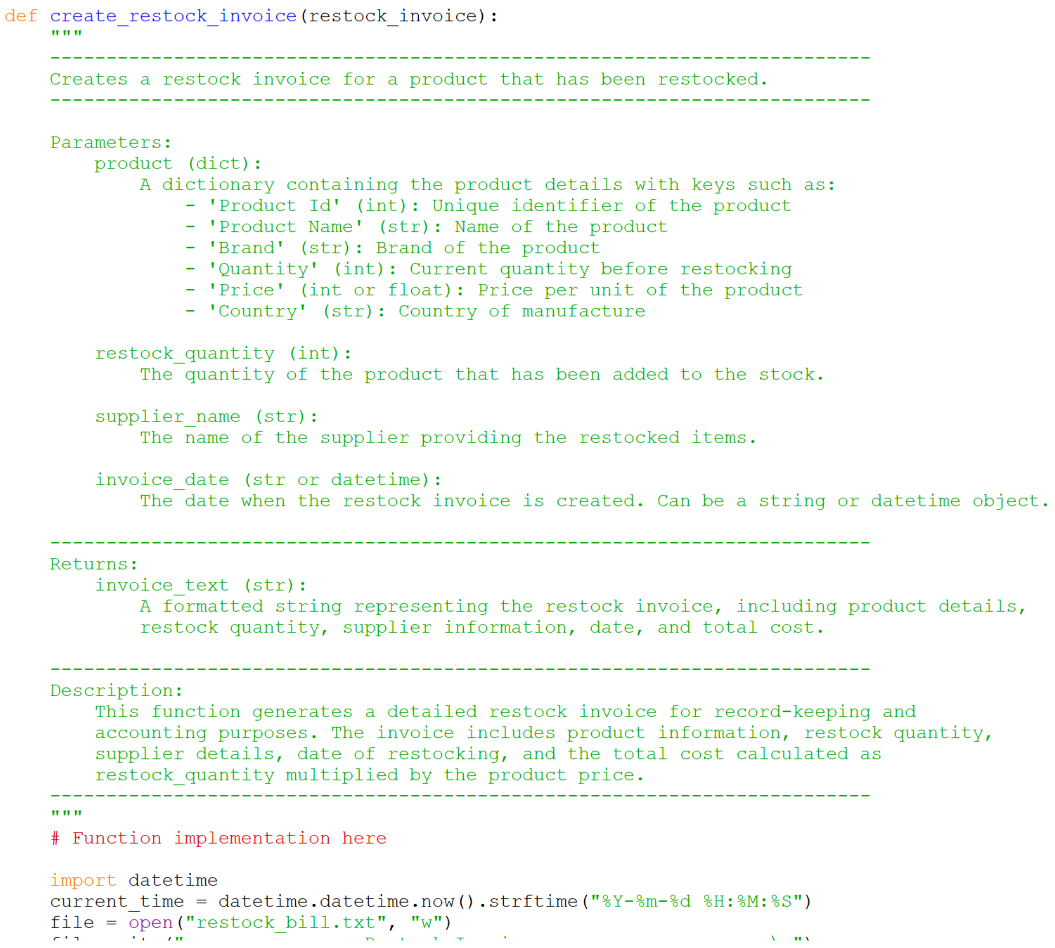
It opens a file with name product.txt for writing. It collects the details of serial number, name, brand, quantity, cost price and origin of each product. It writes all the details into the file in one line separated by commas. Once all products are written, it closes the file to save changes.



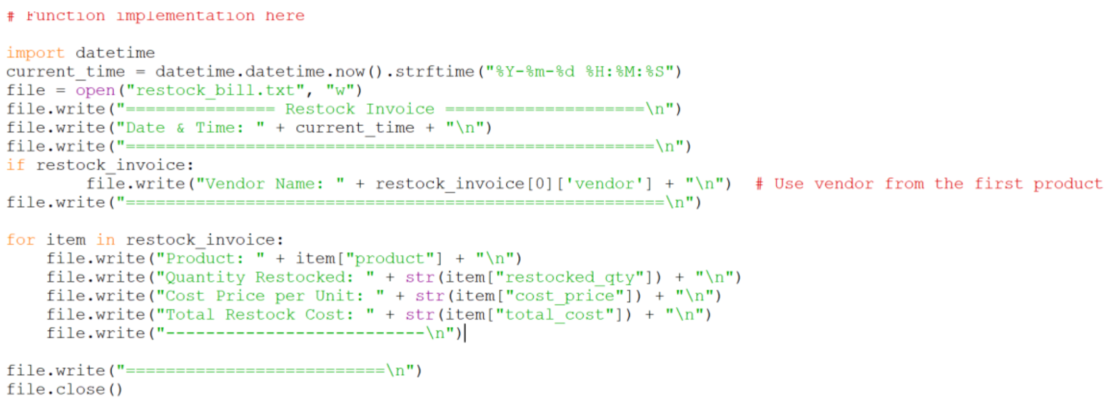
```python
def save_products(filename, products):
    file = open(filename, 'w')
    for p in products:
        line = p['S.N'] + ',' + p['name'] + ',' + p['brand'] + ',' + str(p['quantity']) + ',' + str(p['cost_price']) + ',' + p['origin'] + '\n'
        file.write(line)
    file.close()
```

.

It creates bill with current date and time, a header, vendor name by opening the file called restock_bill.txt for each restock product along with product name, quantity restocked, cost per unit, total cost. After writing all the information, it closes the file.



```python
def create_restock_invoice(restock_invoice):
    """
    -------------------------------------------------------------------------
    Creates a restock invoice for a product that has been restocked.
    -------------------------------------------------------------------------

    Parameters:
        product (dict):
            A dictionary containing the product details with keys such as:
                - 'Product Id' (int): Unique identifier of the product
                - 'Product Name' (str): Name of the product
                - 'Brand' (str): Brand of the product
                - 'Quantity' (int): Current quantity before restocking
                - 'Price' (int or float): Price per unit of the product
                - 'Country' (str): Country of manufacture

        restock_quantity (int):
            The quantity of the product that has been added to the stock.

        supplier_name (str):
            The name of the supplier providing the restocked items.

        invoice_date (str or datetime):
            The date when the restock invoice is created. Can be a string or datetime object.

    -------------------------------------------------------------------------
    Returns:
        invoice_text (str):
            A formatted string representing the restock invoice, including product details,
            restock quantity, supplier information, date, and total cost.

    -------------------------------------------------------------------------
    Description:
        This function generates a detailed restock invoice for record-keeping and
        accounting purposes. The invoice includes product information, restock quantity,
        supplier details, date of restocking, and the total cost calculated as
        restock_quantity multiplied by the product price.
    -------------------------------------------------------------------------
    """
    # Function implementation here

    import datetime
    current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    file = open("restock_bill.txt", "w")
```



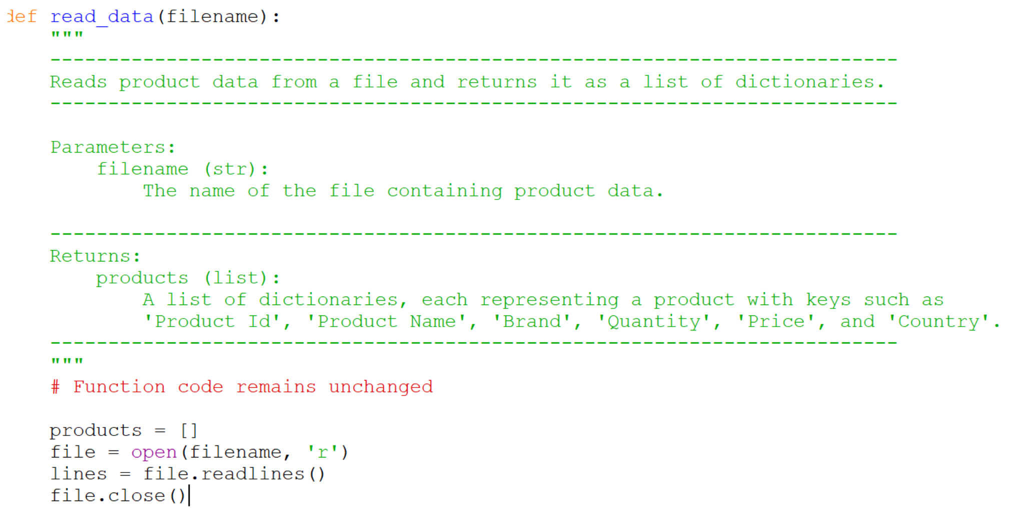
```python
# Function implementation here

import datetime
current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
file = open("restock_bill.txt", "w")
file.write("=============== Restock Invoice ====================\n")
file.write("Date & Time: " + current_time + "\n")
file.write("=====================================================\n")
if restock_invoice:
        file.write("Vendor Name: " + restock_invoice[0]['vendor'] + "\n")  # Use vendor from the first product
file.write("=====================================================\n")

for item in restock_invoice:
    file.write("Product: " + item["product"] + "\n")
    file.write("Quantity Restocked: " + str(item["restocked_qty"]) + "\n")
    file.write("Cost Price per Unit: " + str(item["cost_price"]) + "\n")
    file.write("Total Restock Cost: " + str(item["total_cost"]) + "\n")
    file.write("--------------------------\n")

file.write("==========================\n")
file.close()
```

It shows the restock invoice or restock bill by opening restock_bill.txt file. which as created earlier. Read its contents and when the program runs, it prints full invoice and close the file.

```
ef display_restock_invoice():
    """
    ------------------------------------------------------------------------
    Displays the restock invoice details to the console or user interface.
    ------------------------------------------------------------------------

    Parameters:
        invoice_text (str):
            A formatted string representing the restock invoice. This string
            usually contains product details, restock quantity, supplier information,
            date of restocking, and total cost.

    ------------------------------------------------------------------------
    Returns:
        None
        This function does not return any value. It outputs the invoice text
        directly to the console or display.
    ------------------------------------------------------------------------

    Description:
        This function takes a restock invoice string and prints it in a readable
        format for record-keeping or user confirmation. It helps in verifying
        restock transactions by showing all relevant details clearly.
    ------------------------------------------------------------------------
    """
    # Example implementation
    print(invoice_text)

    print("\n==== Restock Invoice ====")
    print("=========================")
    file = open("restock_bill.txt", "r")
    print(file.read())
    file.close()
```

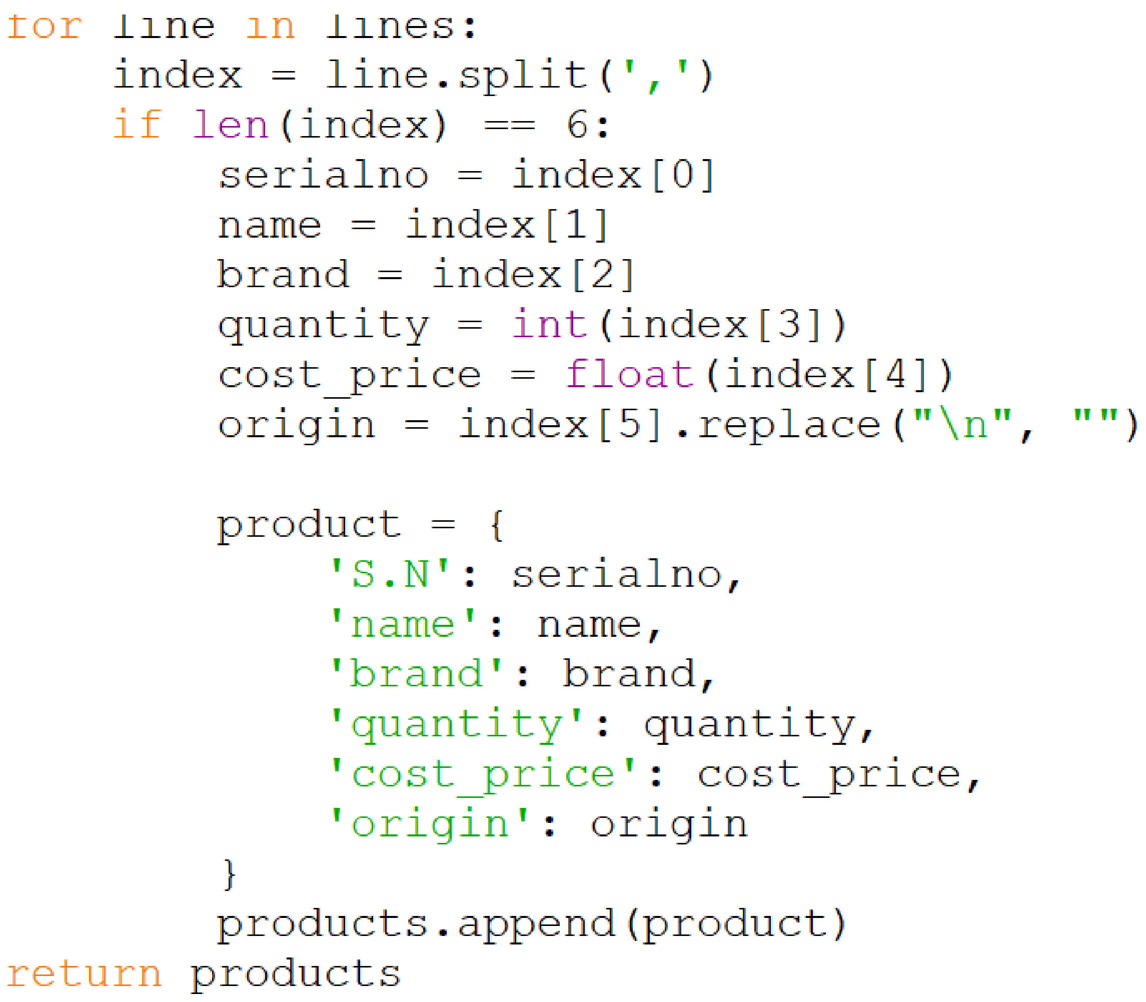
read.py

Open the products.txt file which contains a list of products each written on its own line. It reads the whole file line by line for the data of single products separated by commas into separated parts: serial number, product name, brand quantity (number), cost price (decimal number), origin and removes the new line character(\n) from last item so the data is clean.



```
def read_data(filename):
    """
    -------------------------------------------------------------------------
    Reads product data from a file and returns it as a list of dictionaries.
    -------------------------------------------------------------------------

    Parameters:
        filename (str):
            The name of the file containing product data.

    -------------------------------------------------------------------------
    Returns:
        products (list):
            A list of dictionaries, each representing a product with keys such as
            'Product Id', 'Product Name', 'Brand', 'Quantity', 'Price', and 'Country'.
    -------------------------------------------------------------------------
    """
    # Function code remains unchanged

    products = []
    file = open(filename, 'r')
    lines = file.readlines()
    file.close()
```

It creates the product received by putting all the information into a dictionary with proper labels.



```
for line in lines:
    index = line.split(',')
    if len(index) == 6:
        serialno = index[0]
        name = index[1]
        brand = index[2]
        quantity = int(index[3])
        cost_price = float(index[4])
        origin = index[5].replace("\n", "")

        product = {
            'S.N': serialno,
            'name': name,
            'brand': brand,
            'quantity': quantity,
            'cost_price': cost_price,
            'origin': origin
        }
        products.append(product)
return products
```

All these products dictionaries are added into a big list called products which stores all the product from the file.

At the end, the function returns the full list of products so other part of program can use it.

operations.py

This code helps to manage a product store where the owner can :

- View all the available products
- Sell products to customers (with an offer)
- Restock products and generate an invoice for it.

This part shows all the available products in a neat table on the screen. Each product is shown with its serial number (Id), name, brand, stock left, cost price (original price), selling price (cost price*2), origin. It helps the user quickly see which products are available, how many are in stock and at what price.

```python
import write
def show_producttable(products):
    """
    -------------------------------------------------------------------------
    Displays the list of products in a tabular format.
    -------------------------------------------------------------------------

    Parameters:
        products (list):
            A list of dictionaries, each representing a product.

    -------------------------------------------------------------------------
    Returns:
        None
        This function does not return anything.
    -------------------------------------------------------------------------
    """
    # Function code remains unchanged

    print("Available Products\n")
    print(" S.N  Product              Brand        Stock  C.P       S.P        Origin")
    print("-------------------------------------------------------------------------------")

    for p in products:
        sell_price = p['cost_price'] * 2

        serialno = str(int(p['S.N']))
        if len(serialno) < 6:
            serialno = serialno + ' ' * (6 - len(serialno))

        name = p['name']
        if len(name) < 20:
            name = name + ' ' * (20 - len(name))

        brand = p['brand']
        if len(brand) < 15:
            brand = brand + ' ' * (15 - len(brand))

        stock = str(p['quantity'])
        if len(stock) < 6:
            stock = stock + ' ' * (6 - len(stock))

        cost = str(int(p['cost_price']))
        if len(cost) < 11:
            cost = cost + ' ' * (11 - len(cost))
```

Users enter product id, ask for number of items buy, automatically applies buy 3 get 1 free offer, reduces items from stock. Again, ask for customer to buy more items or not. Return the details of item or not. Return the details of items that are sold, paid for how many items and how many free items did the user get. It makes easy selling and ensures that free items are calculated properly.

Ask the user vendor name who supplies the stock. Ask for product id to restock and number of items to add and calculate total restock cost. Update the restock with new quantities. Records the restock items in the restock invoice with product name, quantity restored, cost per items, total cost, vendor name. Save the new update product list back to file. Display neat bill at the end.

Save the updated product list to a text file so it doesn't disappear. Generate and show a restock invoice in a readable format for future reference.

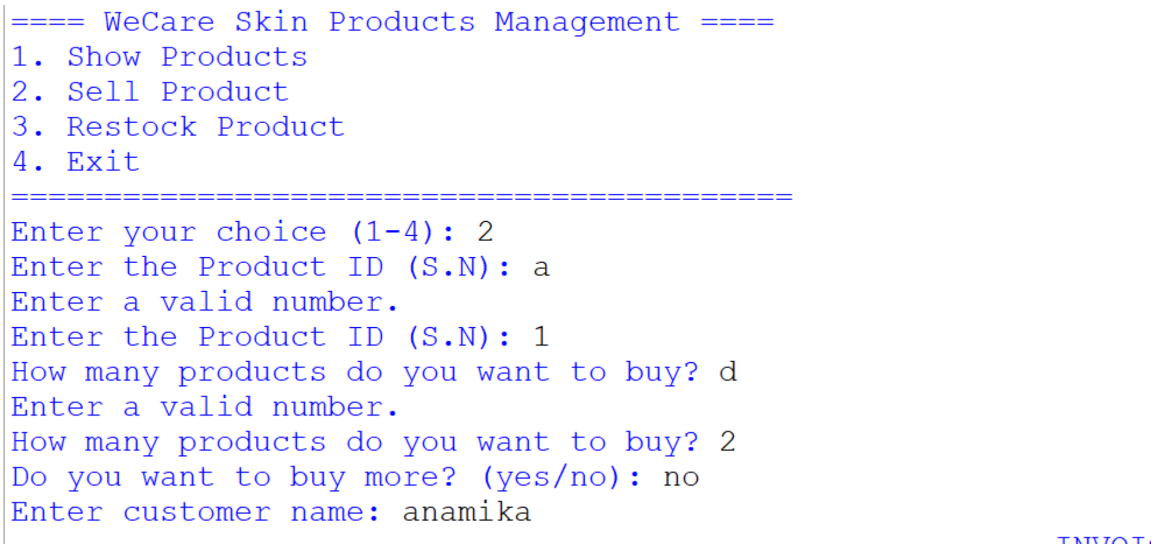
```
            if not found:
                print("Invalid Product ID. Please try again.")

        except ValueError:
            print("Invalid input. Please enter a numeric Product ID.")

        another_restock = input("Do you want to restock another product? (Y/N): ").upper()
        if another_restock != 'Y':
            break

    if restock_invoice:
        write.create_restock_invoice(restock_invoice)
        write.display_restock_invoice()
    else:
        print("\nNo products were restocked, so no restock invoice was created.\n")
```

4.Testing

4.1. Show implementation of try, except. Provide invalid input and show the message

Objectives	To check either it gives message on invalid input or not.
Action	Give the string or alphabet in product id.
Expected result	Message of Invalid input

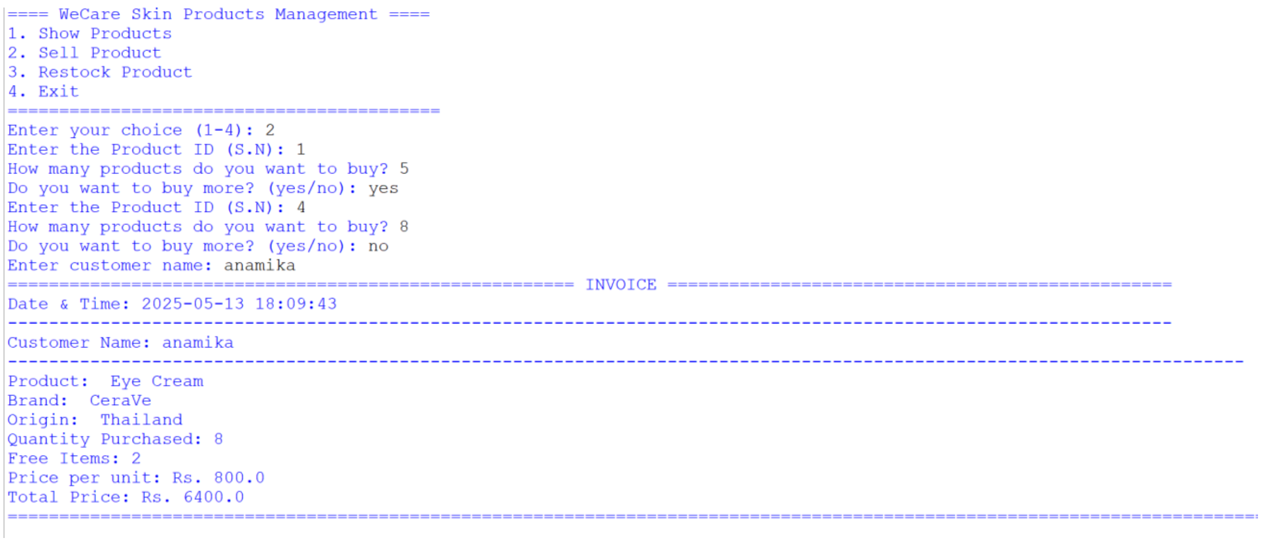
Actual result	Message shown as enter the valid number.
Conclusion	The test was successfully done.



```
==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 2
Enter the Product ID (S.N): a
Enter a valid number.
Enter the Product ID (S.N): 1
How many products do you want to buy? d
Enter a valid number.
How many products do you want to buy? 2
Do you want to buy more? (yes/no): no
Enter customer name: anamika
```

4.2 Selection purchase and sales of products

Objectives	To select purchase and sales of goods.
Action	Enter the product id,number of products and customer name
Expected result	Invoice created with customer name and other labels
Actual result	Invoice was created
Conclusion	The test was successful


```
==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 2
Enter the Product ID (S.N): 1
How many products do you want to buy? 5
Do you want to buy more? (yes/no): yes
Enter the Product ID (S.N): 4
How many products do you want to buy? 8
Do you want to buy more? (yes/no): no
Enter customer name: anamika
======================================================= INVOICE ==============================================
Date & Time: 2025-05-13 18:09:43
---------------------------------------------------------------------------------------------------------------
Customer Name: anamika
----------------------------------------------------------------------------------------------------------------------
Product:  Eye Cream
Brand:  CeraVe
Origin:  Thailand
Quantity Purchased: 8
Free Items: 2
Price per unit: Rs. 800.0
Total Price: Rs. 6400.0
=========================================================================================================================
```

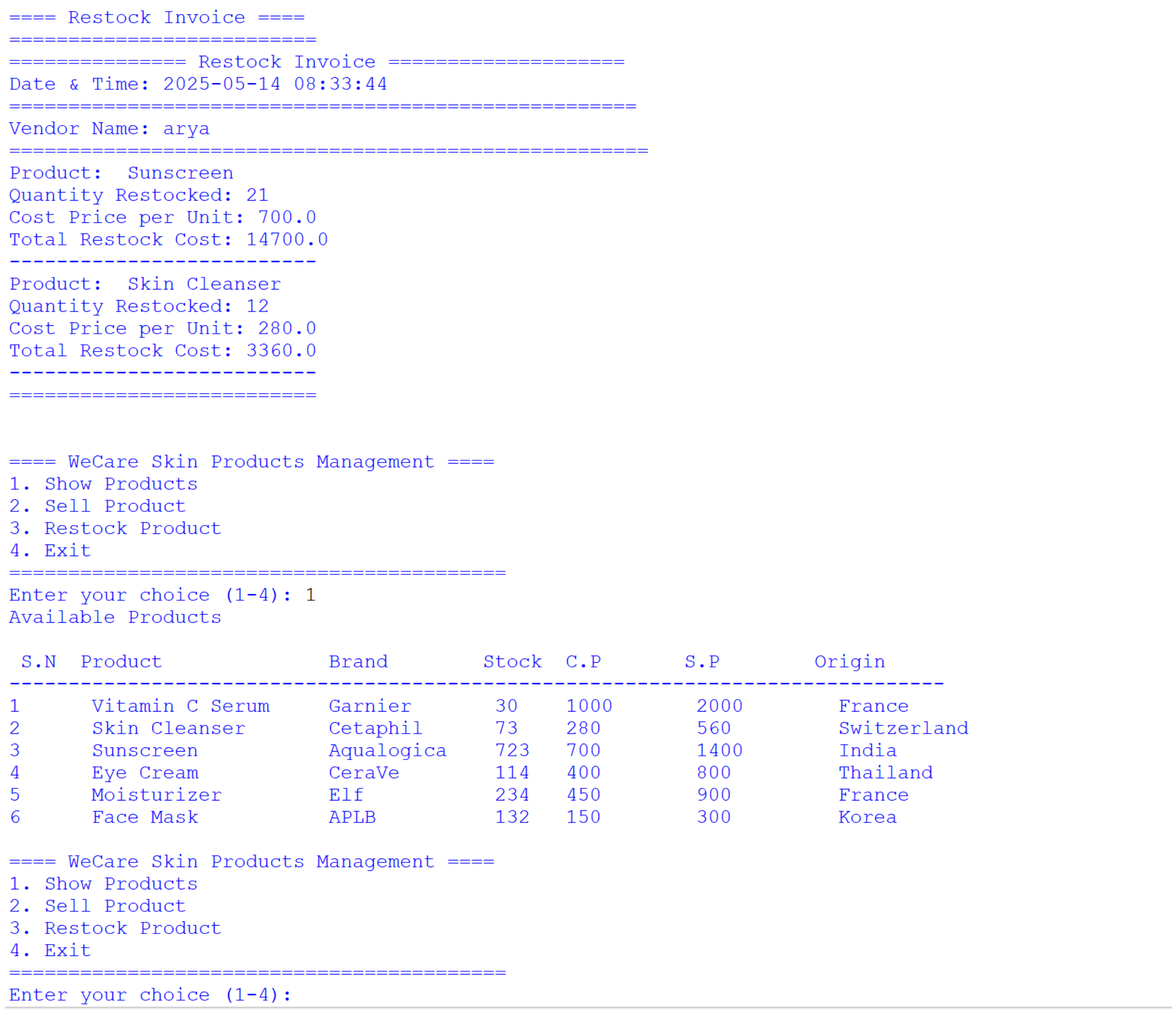
4.3 File generation of purchase of products (purchasing multiple products)

Show complete purchase process

Show output in the shell as well

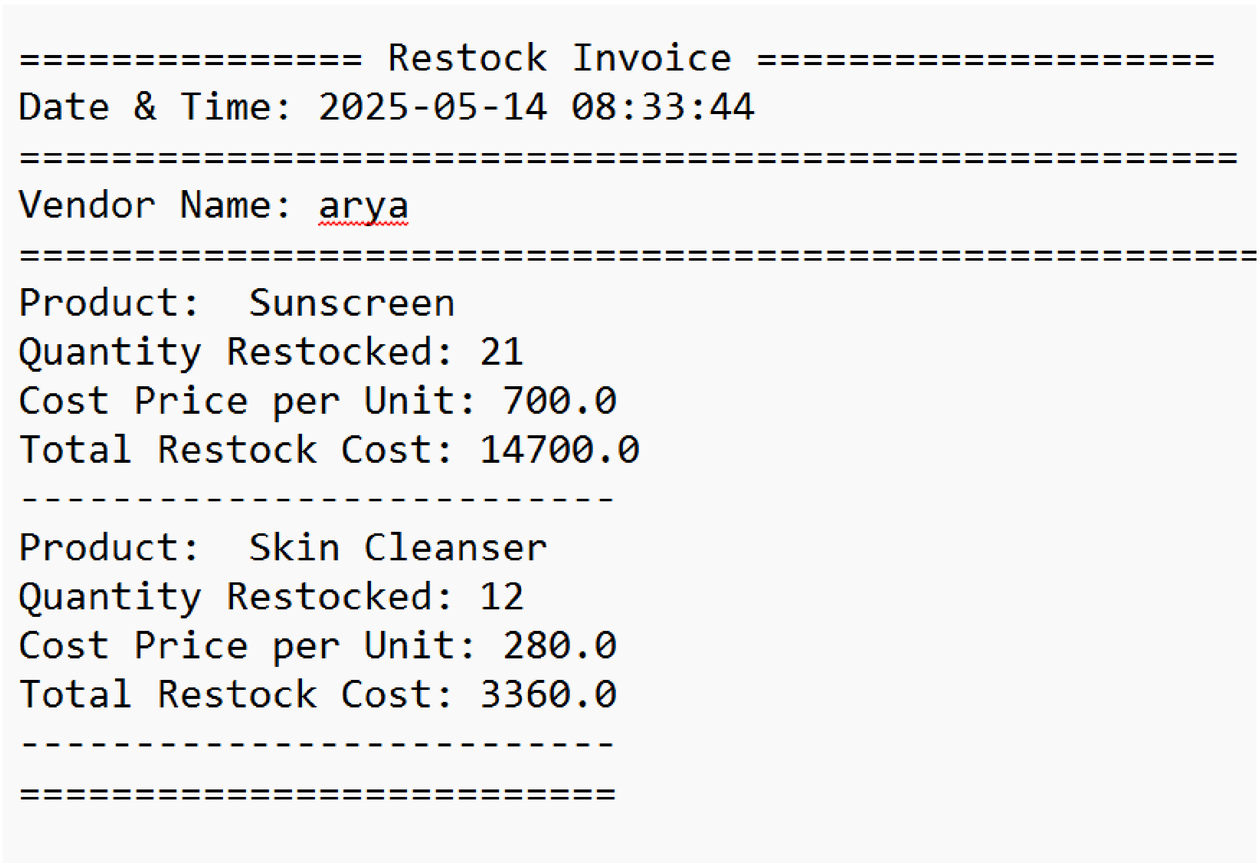
Finally show the purchased products details in text file

Objectives	To generate the invoice of bill and complete purchase process
Action	Enter the option 3,enter product id and quantity to restock and created the restock bill
Expected result	The invoice is created, and the text file is created and update the value.
Actual result	The invoice was displayed, and the text file of bill is created.
Conclusion	The test was successfully done.



```
==== Restock Invoice ====
==========================
=============== Restock Invoice ====================
Date & Time: 2025-05-14 08:33:44
=====================================================
Vendor Name: arya
======================================================
Product:  Sunscreen
Quantity Restocked: 21
Cost Price per Unit: 700.0
Total Restock Cost: 14700.0
--------------------------
Product:  Skin Cleanser
Quantity Restocked: 12
Cost Price per Unit: 280.0
Total Restock Cost: 3360.0
--------------------------
==========================


==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 1
Available Products

 S.N  Product               Brand        Stock  C.P       S.P        Origin
-------------------------------------------------------------------------------
1      Vitamin C Serum      Garnier       30    1000       2000        France
2      Skin Cleanser        Cetaphil      73    280        560         Switzerland
3      Sunscreen            Aqualogica    723   700        1400        India
4      Eye Cream            CeraVe        114   400        800         Thailand
5      Moisturizer          Elf           234   450        900         France
6      Face Mask            APLB          132   150        300         Korea

==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4):
```

```
==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 1
Available Products

 S.N  Product                Brand        Stock  C.P       S.P        Origin
------------------------------------------------------------------------------
1      Vitamin C Serum     Garnier       30    1000        2000        France
2      Skin Cleanser       Cetaphil      61    280         560         Switzerland
3      Sunscreen           Aqualogica    702   700         1400        India
4      Eye Cream           CeraVe        114   400         800         Thailand
5      Moisturizer         Elf           234   450         900         France
6      Face Mask           APLB          132   150         300         Korea

==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 3
Enter vendor name: arya
Enter the Product ID to restock: 3
Current quantity of ' Sunscreen ': 702
Quantity to add: 21
 Sunscreen  restocked successfully. New quantity: 723
Do you want to restock another product? (Y/N): Y
Enter the Product ID to restock: 2
Current quantity of ' Skin Cleanser': 61
Quantity to add: 12
 Skin Cleanser restocked successfully. New quantity: 73
Do you want to restock another product? (Y/N): N
```



```
=============== Restock Invoice ====================
Date & Time: 2025-05-14 08:33:44
=====================================================
Vendor Name: arya
======================================================
Product:  Sunscreen
Quantity Restocked: 21
Cost Price per Unit: 700.0
Total Restock Cost: 14700.0
--------------------------
Product:  Skin Cleanser
Quantity Restocked: 12
Cost Price per Unit: 280.0
Total Restock Cost: 3360.0
--------------------------
==========================
```

4.4 File generation of sales process of products (Selling multiple products)

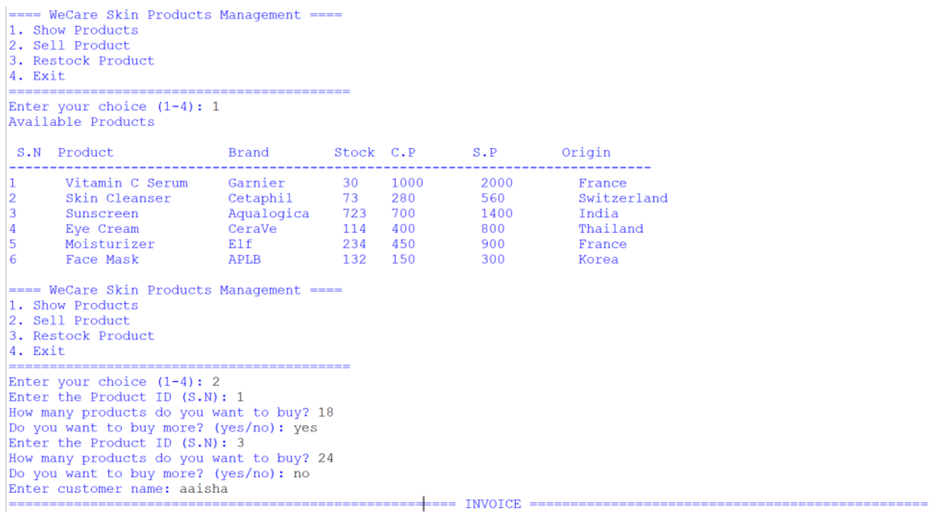
Show complete sales process of products

Show output in the shell as well

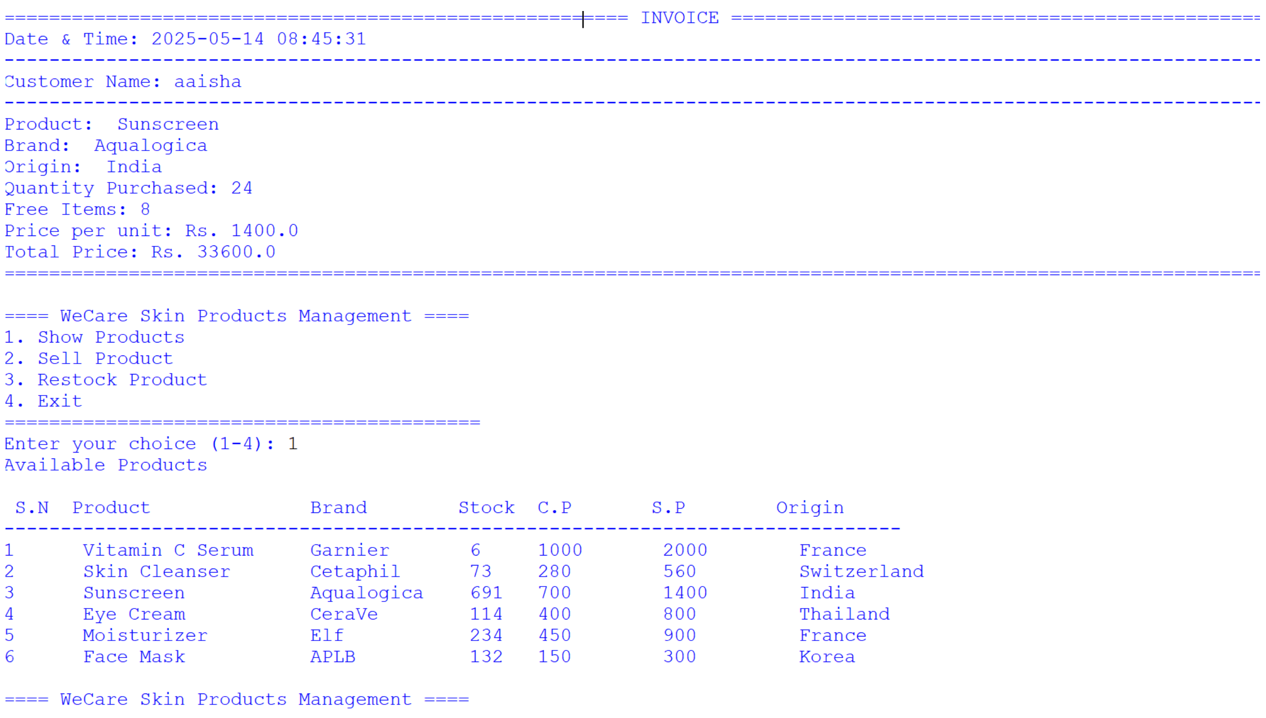
Finally show the sold products details in text file

Objectives	To generate invoice of sell product in text file as well
Action	Choose the option 2 and enter the product id and details for invoice

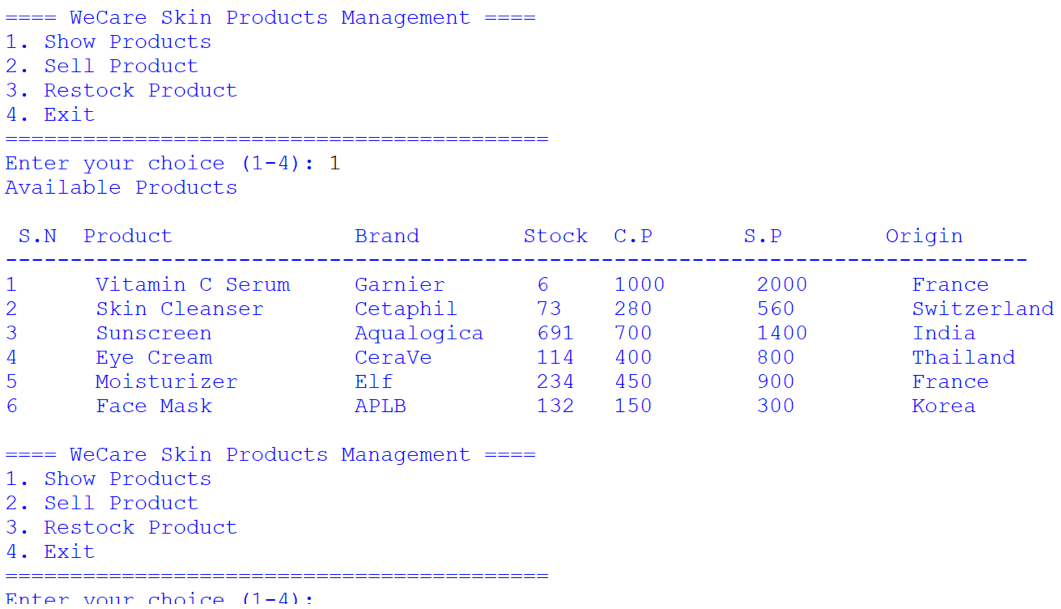
Expected result	The invoice is created with in screen and text file.
Actual result	The invoice was created
Conclusion	The test was done successfully.



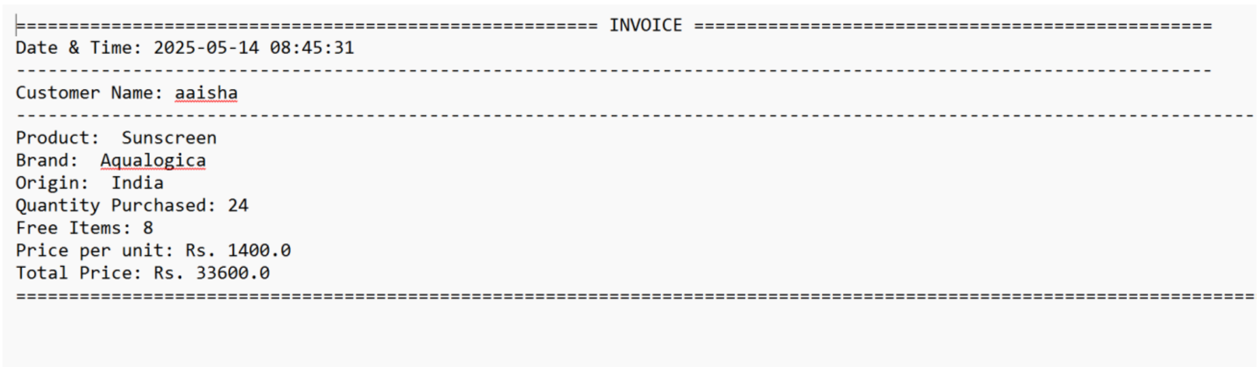
```
==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 1
Available Products

 S.N  Product              Brand        Stock  C.P       S.P        Origin
-------------------------------------------------------------------------------
1      Vitamin C Serum     Garnier       30    1000       2000        France
2      Skin Cleanser       Cetaphil      73    280        560         Switzerland
3      Sunscreen           Aqualogica    723   700        1400        India
4      Eye Cream           CeraVe        114   400        800         Thailand
5      Moisturizer         Elf           234   450        900         France
6      Face Mask           APLB          132   150        300         Korea

==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 2
Enter the Product ID (S.N): 1
How many products do you want to buy? 18
Do you want to buy more? (yes/no): yes
Enter the Product ID (S.N): 3
How many products do you want to buy? 24
Do you want to buy more? (yes/no): no
Enter customer name: aaisha
======================================================= INVOICE ===============================================
```



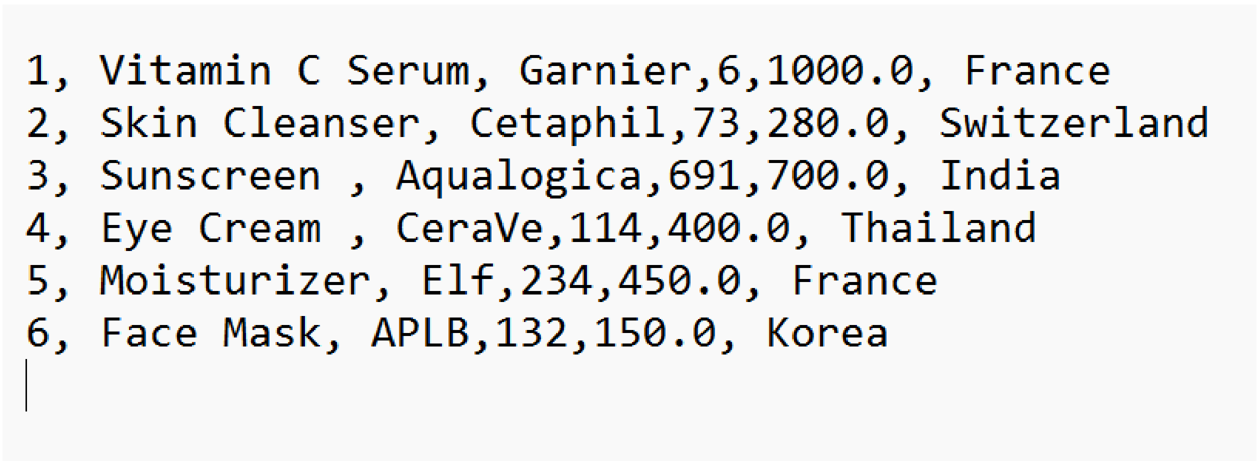
```
======================================================= INVOICE ===============================================
Date & Time: 2025-05-14 08:45:31
------------------------------------------------------------------------------------------------------------
Customer Name: aaisha
------------------------------------------------------------------------------------------------------------
Product:  Sunscreen
Brand:  Aqualogica
Origin:  India
Quantity Purchased: 24
Free Items: 8
Price per unit: Rs. 1400.0
Total Price: Rs. 33600.0
============================================================================================================

==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 1
Available Products

 S.N  Product              Brand        Stock  C.P       S.P        Origin
-------------------------------------------------------------------------------
1      Vitamin C Serum     Garnier       6     1000       2000        France
2      Skin Cleanser       Cetaphil      73    280        560         Switzerland
3      Sunscreen           Aqualogica    691   700        1400        India
4      Eye Cream           CeraVe        114   400        800         Thailand
5      Moisturizer         Elf           234   450        900         France
6      Face Mask           APLB          132   150        300         Korea

==== WeCare Skin Products Management ====
```



```
==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 1
Available Products

 S.N  Product              Brand        Stock  C.P       S.P        Origin
--------------------------------------------------------------------------------
1      Vitamin C Serum     Garnier       6     1000       2000        France
2      Skin Cleanser       Cetaphil      73    280        560         Switzerland
3      Sunscreen           Aqualogica    691   700        1400        India
4      Eye Cream           CeraVe        114   400        800         Thailand
5      Moisturizer         Elf           234   450        900         France
6      Face Mask           APLB          132   150        300         Korea

==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4):
```



```
======================================================= INVOICE ==================================================
Date & Time: 2025-05-14 08:45:31
------------------------------------------------------------------------------------------------------------------
Customer Name: aaisha
------------------------------------------------------------------------------------------------------------------
Product:  Sunscreen
Brand:  Aqualogica
Origin:  India
Quantity Purchased: 24
Free Items: 8
Price per unit: Rs. 1400.0
Total Price: Rs. 33600.0
==================================================================================================================
```



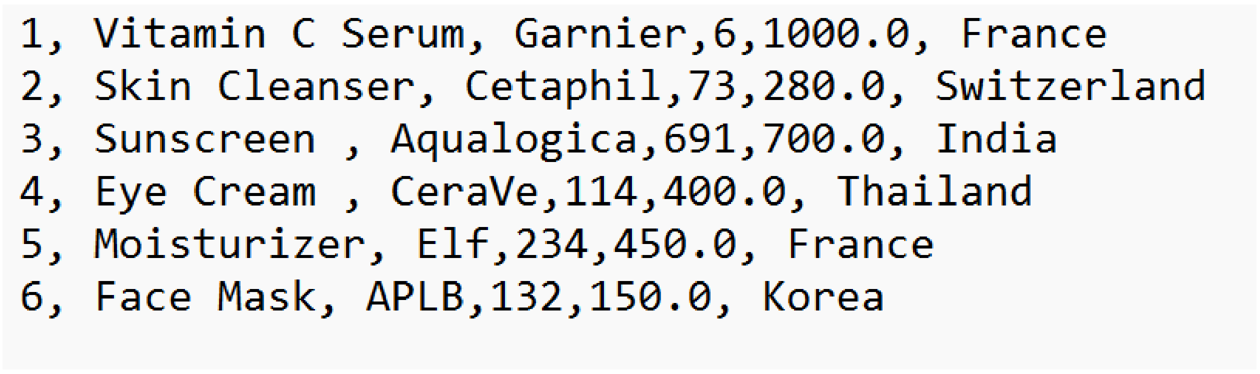
```
1, Vitamin C Serum, Garnier,6,1000.0, France
2, Skin Cleanser, Cetaphil,73,280.0, Switzerland
3, Sunscreen , Aqualogica,691,700.0, India
4, Eye Cream , CeraVe,114,400.0, Thailand
5, Moisturizer, Elf,234,450.0, France
6, Face Mask, APLB,132,150.0, Korea
```

4.5 Show the update in the stock of products

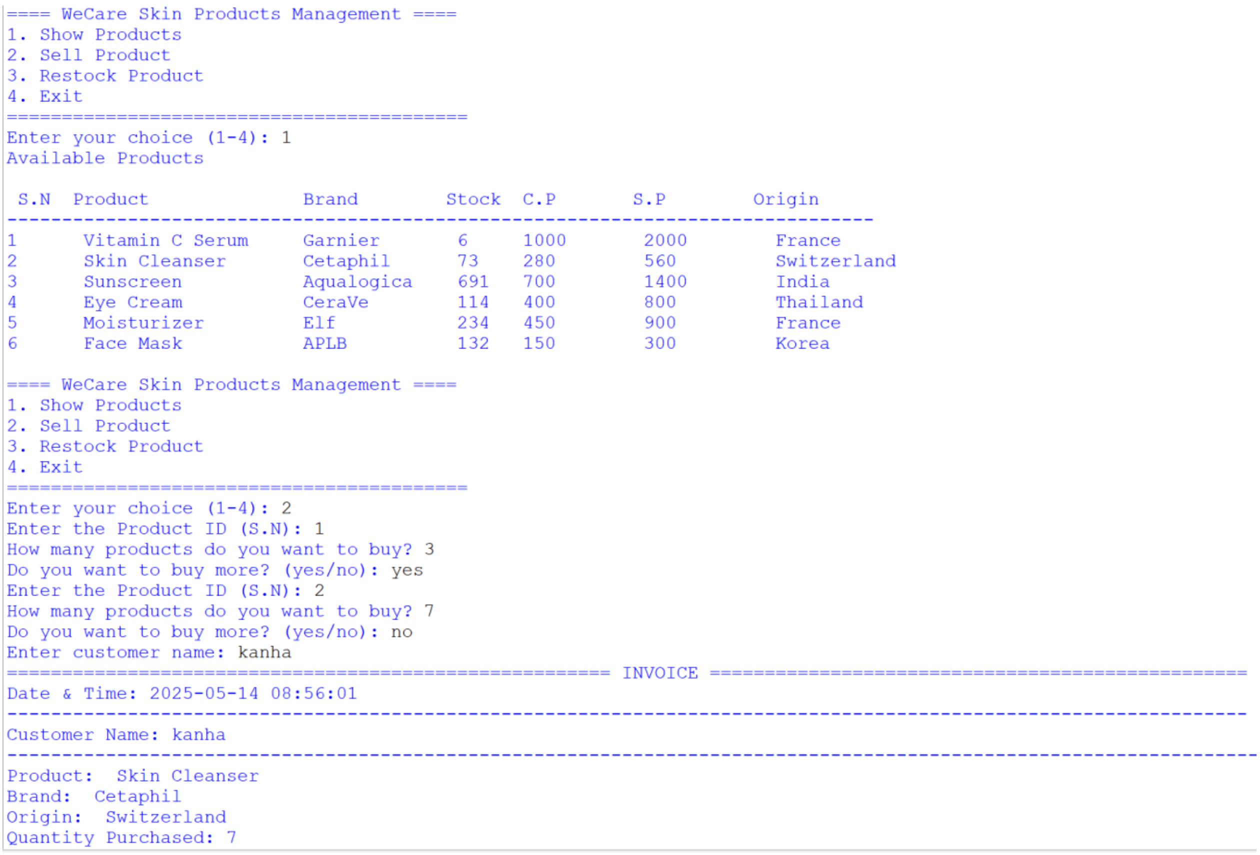
Show the quantity being deducted while purchasing the product (Update should be reflected in a txt.file as well)

Show the quantity being added while selling the product (Update should be reflected in a txt.file as well)

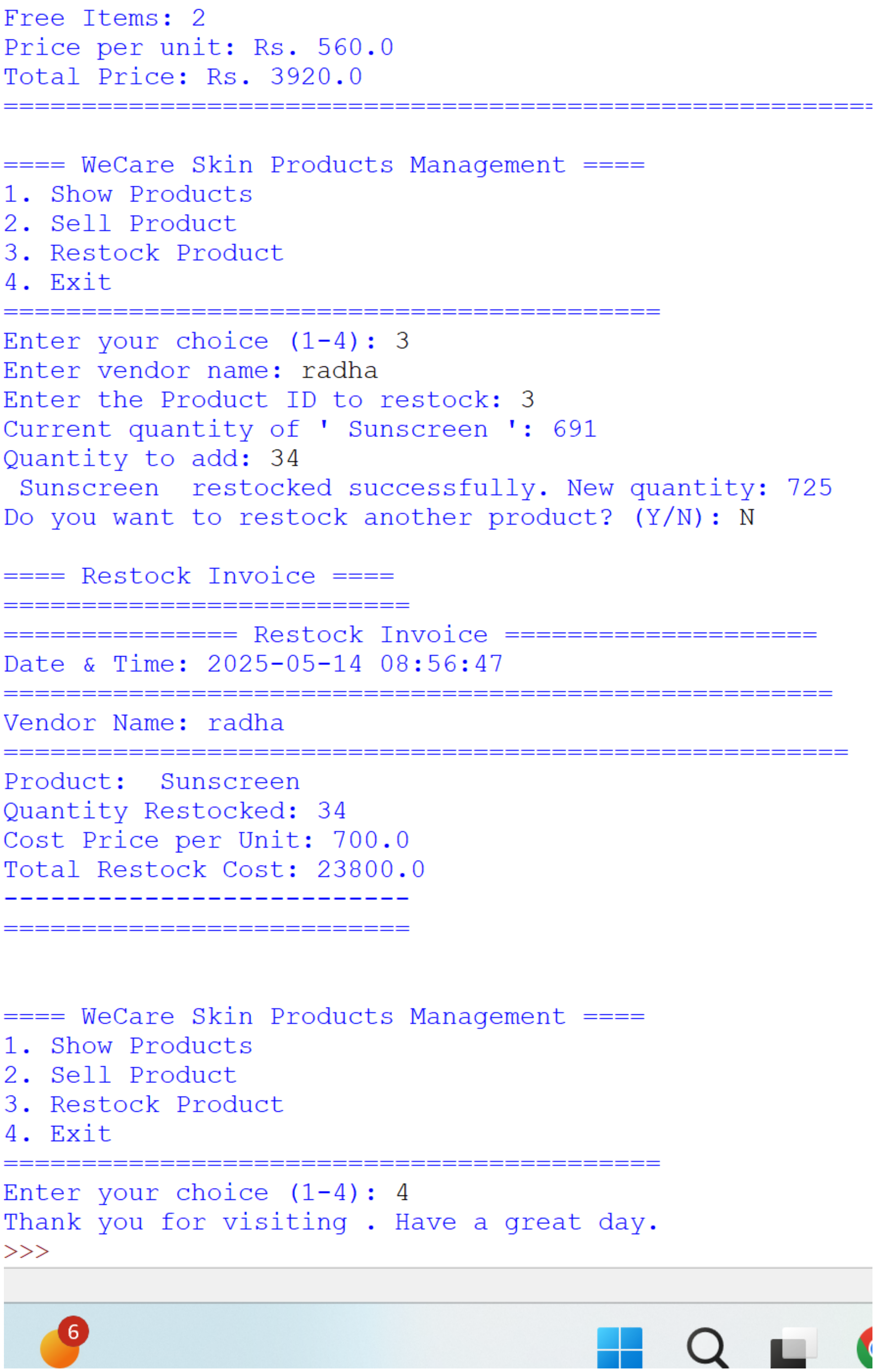
Objectives	To show the update in the stock of products.
Action	To enter option 2 and 3 and update the restock in text file
Expected result	Updated quantity is in text file
Actual result	The text file was updated
Conclusion	The test was done successfully.

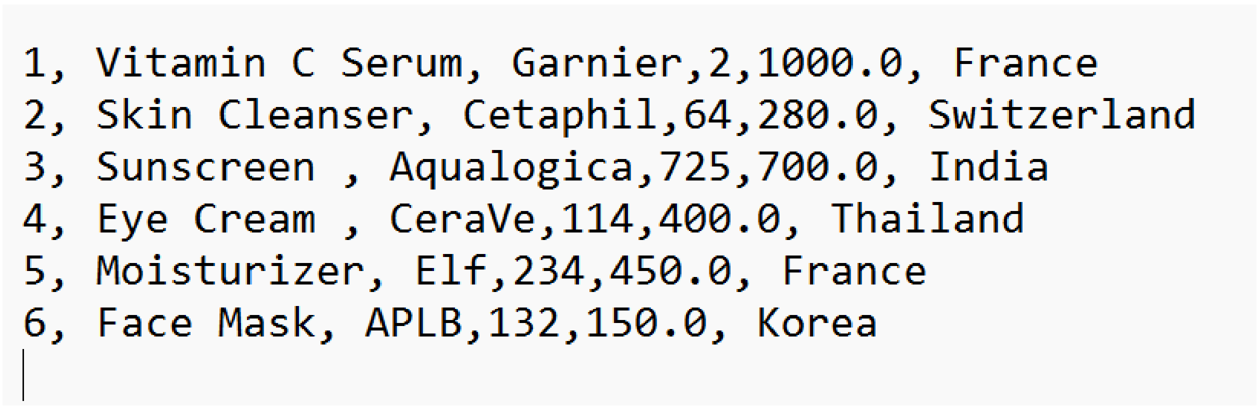


```
1, Vitamin C Serum, Garnier,6,1000.0, France
2, Skin Cleanser, Cetaphil,73,280.0, Switzerland
3, Sunscreen , Aqualogica,691,700.0, India
4, Eye Cream , CeraVe,114,400.0, Thailand
5, Moisturizer, Elf,234,450.0, France
6, Face Mask, APLB,132,150.0, Korea
```



```
==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 1
Available Products

 S.N  Product                Brand        Stock  C.P       S.P        Origin
-------------------------------------------------------------------------------
1      Vitamin C Serum      Garnier        6     1000       2000        France
2      Skin Cleanser        Cetaphil       73    280        560         Switzerland
3      Sunscreen            Aqualogica     691   700        1400        India
4      Eye Cream            CeraVe         114   400        800         Thailand
5      Moisturizer          Elf            234   450        900         France
6      Face Mask            APLB           132   150        300         Korea

==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 2
Enter the Product ID (S.N): 1
How many products do you want to buy? 3
Do you want to buy more? (yes/no): yes
Enter the Product ID (S.N): 2
How many products do you want to buy? 7
Do you want to buy more? (yes/no): no
Enter customer name: kanha
======================================================= INVOICE =================================================
Date & Time: 2025-05-14 08:56:01
-----------------------------------------------------------------------------------------------------------------
Customer Name: kanha
------------------------------------------------------------------------------------------------------------------
Product:  Skin Cleanser
Brand:  Cetaphil
Origin:  Switzerland
Quantity Purchased: 7
```


```
Free Items: 2
Price per unit: Rs. 560.0
Total Price: Rs. 3920.0
=======================================================

==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 3
Enter vendor name: radha
Enter the Product ID to restock: 3
Current quantity of ' Sunscreen ': 691
Quantity to add: 34
 Sunscreen  restocked successfully. New quantity: 725
Do you want to restock another product? (Y/N): N

==== Restock Invoice ====
==========================
=============== Restock Invoice ====================
Date & Time: 2025-05-14 08:56:47
=====================================================
Vendor Name: radha
======================================================
Product:  Sunscreen
Quantity Restocked: 34
Cost Price per Unit: 700.0
Total Restock Cost: 23800.0
--------------------------
==========================


==== WeCare Skin Products Management ====
1. Show Products
2. Sell Product
3. Restock Product
4. Exit
==========================================
Enter your choice (1-4): 4
Thank you for visiting . Have a great day.
>>>
```



```
1, Vitamin C Serum, Garnier,2,1000.0, France
2, Skin Cleanser, Cetaphil,64,280.0, Switzerland
3, Sunscreen , Aqualogica,725,700.0, India
4, Eye Cream , CeraVe,114,400.0, Thailand
5, Moisturizer, Elf,234,450.0, France
6, Face Mask, APLB,132,150.0, Korea
```

5.Conclusion

This whole project is done for the real-life application of python code to make a product store. Every line of code used here in The WeCare Skin Products Retail Billing System project successfully addresses the operational needs of a small-scale beauty and skincare retail business. By implementing core functionalities such as product data management, customer billing with a "Buy 3 Get 1 Free" promotional policy, VAT-inclusive invoice generation, and stock replenishment tracking, the system enhances both efficiency and customer satisfaction.

Developed using Python, the system demonstrates strong modularity by organizing code into separate files for reading, writing, operations, and main execution. It ensures data persistence through file handling and enables the store administrator to perform sales and restocking seamlessly. The use of clear menus, input validation, and structured invoice output makes the system both user-friendly and reliable.

In conclusion, this project not only fulfills the technical objectives but also reflects practical understanding of inventory control, sales automation, and customer-focused design. With further expansion—such as GUI integration or database support—this system has the potential to evolve into a more scalable and robust retail management solution for WeCare or similar businesses.

6.Appendix

Main

```
from read import read_data
from write import save_products, create_restock_invoice, display_restock_invoice
from operations import show_producttable, selling_products, restock_product
import datetime

def main():
    """
    ---------------------------------------------------------------------
    Main function to run the WeCare Skin Products Management system.
    ---------------------------------------------------------------------

    Description:
        This function serves as the entry point for the application. It loads
        product data, displays a menu to the user, and handles user choices
        for showing products, selling products, restocking, and exiting the
        program. It also manages invoice creation and product stock updates.

    ---------------------------------------------------------------------
    Parameters:
        None

    ---------------------------------------------------------------------
    Returns:
        None
        This function does not return anything.
    ---------------------------------------------------------------------
    """


    filename = "products.txt"

    products = read_data(filename)

    """

This function reads data from a file,

processes each line to extract product details,

and returns a list of product dictionaries.

"""






    while True:

        print("\n==== WeCare Skin Products Management ====")

        print("1. Show Products")

        print("2. Sell Product")

        print("3. Restock Product")

        print("4. Exit")

        print("==========================================")


        choice = input("Enter your choice (1-4): ")


        if choice == "1":

            show_producttable(products)
        elif choice == "2":
            sold_product, bought_qty, free_items = selling_products(products)

            customer_name = input("Enter customer name: ")

            total_price = bought_qty * sold_product['cost_price'] * 2
            current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

            invoice_text = []  # Initialize invoice_text as a list


invoice_text.append("================================================
=======                                                          INVOICE
=================================================\n")
            invoice_text.append("Date & Time: " + current_time + "\n")
            invoice_text.append("-----------------------------------------------------------------------
------------------------------------\n")
            invoice_text.append("Customer Name: " + customer_name + "\n")
            invoice_text.append("-----------------------------------------------------------------------
-------------------------------------------\n")
            invoice_text.append("Product: " + sold_product['name'] + "\n")
            invoice_text.append("Brand: " + sold_product['brand'] + "\n")
            invoice_text.append("Origin: " + sold_product['origin'] + "\n")
            invoice_text.append("Quantity Purchased: " + str(bought_qty) + "\n")
            invoice_text.append("Free Items: " + str(free_items) + "\n")
```

```
            invoice_text.append("Price per unit: Rs. " + str(sold_product['cost_price'] * 2) +
"\n")
            invoice_text.append("Total Price: Rs. " + str(total_price) + "\n")

invoice_text.append("================================================
================================================================
==============\n")

            # Show on screen
            for line in invoice_text:
                print(line, end='')

            # Save to file
            with open("invoice.txt", "w") as invoice_file:
                invoice_file.write("".join(invoice_text))

            # Save product stock update
            save_products(filename, products)

        elif choice == "3":
            restock_product(filename, products)

        elif choice == "4":
            print("Thank you for visiting . Have a great day.")
            break
        else:
            print("Invalid choice. Please enter a number from 1 to 4.")


main()
```

Write

```
def save_products(filename, products):
    """
    ----------------------------------------------------------------------
    Saves the updated list of products to the specified file.
    ----------------------------------------------------------------------

    Parameters:
        filename (str):
            The name of the file where product data will be saved.
        products (list):
            A list of dictionaries, each representing a product.

    ----------------------------------------------------------------------
    Returns:
        None
        This function does not return anything.
    ----------------------------------------------------------------------
    """
    # Function code remains unchanged


    file = open(filename, 'w')

    for p in products:

        line = p['S.N'] + ',' + p['name'] + ',' + p['brand'] + ',' + str(p['quantity']) + ',' +
str(p['cost_price']) + ',' + p['origin'] + '\n'

        file.write(line)

    file.close()


def create_restock_invoice(restock_invoice):

    """

    ----------------------------------------------------------------------

    Creates a restock invoice for a product that has been restocked.

    ----------------------------------------------------------------------


    Parameters:

        product (dict):

            A dictionary containing the product details with keys such as:

                - 'Product Id' (int): Unique identifier of the product

                - 'Product Name' (str): Name of the product

                - 'Brand' (str): Brand of the product

                - 'Quantity' (int): Current quantity before restocking

                - 'Price' (int or float): Price per unit of the product

                - 'Country' (str): Country of manufacture
        restock_quantity (int):
            The quantity of the product that has been added to the stock.

        supplier_name (str):
            The name of the supplier providing the restocked items.

        invoice_date (str or datetime):
            The date when the restock invoice is created. Can be a string or datetime object.

    ---------------------------------------------------------------------
    Returns:
        invoice_text (str):
            A formatted string representing the restock invoice, including product details,
            restock quantity, supplier information, date, and total cost.

    ---------------------------------------------------------------------
    Description:
        This function generates a detailed restock invoice for record-keeping and
        accounting purposes. The invoice includes product information, restock quantity,
        supplier details, date of restocking, and the total cost calculated as
        restock_quantity multiplied by the product price.
    ---------------------------------------------------------------------
    """
    # Function implementation here
```

```
    import datetime
    current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    file = open("restock_bill.txt", "w")
    file.write("=============== Restock Invoice ====================\n")
    file.write("Date & Time: " + current_time + "\n")

file.write("=====================================================\n")
    if restock_invoice:
            file.write("Vendor Name: " + restock_invoice[0]['vendor'] + "\n")  # Use vendor
from the first product

file.write("=====================================================\n")

    for item in restock_invoice:
        file.write("Product: " + item["product"] + "\n")
        file.write("Quantity Restocked: " + str(item["restocked_qty"]) + "\n")
        file.write("Cost Price per Unit: " + str(item["cost_price"]) + "\n")
        file.write("Total Restock Cost: " + str(item["total_cost"]) + "\n")
        file.write("--------------------------\n")

    file.write("==========================\n")
    file.close()

def display_restock_invoice():
    """
```

```
------------------------------------------------------------------------
Displays the restock invoice details to the console or user interface.
------------------------------------------------------------------------

Parameters:
    invoice_text (str):
        A formatted string representing the restock invoice. This string
        usually contains product details, restock quantity, supplier information,
        date of restocking, and total cost.

------------------------------------------------------------------------
Returns:
    None
    This function does not return any value. It outputs the invoice text
    directly to the console or display.
------------------------------------------------------------------------

Description:
    This function takes a restock invoice string and prints it in a readable
    format for record-keeping or user confirmation. It helps in verifying
    restock transactions by showing all relevant details clearly.
------------------------------------------------------------------------
"""
# Example implementation
print(invoice_text)
```

```
    print("\n==== Restock Invoice ====")

    print("=========================")

    file = open("restock_bill.txt", "r")

    print(file.read())

    file.close()
```

Operation

```
import write

def show_producttable(products):

    """

    ----------------------------------------------------------------------

    Displays the list of products in a tabular format.

    ----------------------------------------------------------------------



    Parameters:

        products (list):

            A list of dictionaries, each representing a product.



    ----------------------------------------------------------------------

    Returns:

        None

        This function does not return anything.

    ----------------------------------------------------------------------

    """
```

```
# Function code remains unchanged


print("Available Products\n")
print(" S.N  Product              Brand        Stock  C.P       S.P        Origin")
print("-------------------------------------------------------------------------------")


for p in products:
    sell_price = p['cost_price'] * 2


    serialno = str(int(p['S.N']))
    if len(serialno) < 6:
        serialno = serialno + ' ' * (6 - len(serialno))


    name = p['name']
    if len(name) < 20:
        name = name + ' ' * (20 - len(name))


    brand = p['brand']
    if len(brand) < 15:
        brand = brand + ' ' * (15 - len(brand))


    stock = str(p['quantity'])
    if len(stock) < 6:
        stock = stock + ' ' * (6 - len(stock))
        cost = str(int(p['cost_price']))
        if len(cost) < 11:
            cost = cost + ' ' * (11 - len(cost))

        selling = str(int(sell_price))
        if len(selling) < 11:
            selling = selling + ' ' * (11 - len(selling))

        origin = p['origin']

        print(serialno + name + brand + stock + cost + selling + origin)

def selling_products(products):
    """
    ---------------------------------------------------------------------
    Handles the process of selling a product to a customer.
    ---------------------------------------------------------------------

    Parameters:
        products (list):
            A list of dictionaries, each representing a product.

    ---------------------------------------------------------------------
    Returns:
        sold_product (dict):
```

```
        The dictionary of the product that was sold.

    bought_qty (int):

        The quantity of the product purchased.

    free_items (int):

        The number of free items given with the purchase.

----------------------------------------------------------------------

"""

# Function code remains unchanged




while True:

    user_input = input("Enter the Product ID (S.N): ")

    if user_input.isdigit():

        found = False

        for product in products:

            if product['S.N'] == user_input:

                found = True

                while True:

                    qty_input = input("How many products do you want to buy? ")

                    if qty_input.isdigit():

                        total_needed = int(qty_input)

                        free_items = total_needed // 3

                        total_with_offer = total_needed + free_items

                        if total_with_offer <= product['quantity']:

                            product['quantity'] -= total_with_offer
```

```
                                # Ask if user wants to buy more
                                more = input("Do you want to buy more? (yes/no): ").lower()
                                if more != 'yes':
                                    #  Only return when user finishes shopping
                                    return product, total_needed, free_items
                                else:
                                    # Continue the loop again for another product
                                    break
                            else:
                                print("Not enough stock. Try a smaller quantity.")
                        else:
                            print("Enter a valid number.")
                    break
            if not found:
                print("Product ID not found.")
        else:
            print("Enter a valid number.")


def restock_product(filename, products):
    """
    ----------------------------------------------------------------------
    Handles the process of restocking a product and updates the product file.
    ----------------------------------------------------------------------
```

```
Parameters:
    filename (str):
        The name of the file where product data is stored.
    products (list):
        A list of dictionaries, each representing a product.

----------------------------------------------------------------------
Returns:
    None
    This function does not return anything.
----------------------------------------------------------------------
"""
# Function code remains unchanged

restock_invoice = []

vendor = input("Enter vendor name: ")  # Ask vendor name once at the beginning

while True:
    try:
        restock_id = int(input("Enter the Product ID to restock: "))
        found = False

        for p in products:
            if int(p['S.N']) == restock_id:
                found = True
                current_qty = p['quantity']
                print("Current quantity of '" + p['name'] + "': " + str(current_qty))

                added_quantity = input("Quantity to add: ")
                if added_quantity.isdigit():
                    added_quantity = int(added_quantity)
                    p['quantity'] += added_quantity

                    cost_price = p['cost_price']
                    total_cost = added_quantity * cost_price

                    restock_invoice.append({
                        "product": p['name'],
                        "restocked_qty": added_quantity,
                        "cost_price": cost_price,
                        "total_cost": total_cost,
                        "vendor": vendor  # Add vendor to invoice
                    })

                    print(p['name'] + " restocked successfully. New quantity: " +
str(p['quantity']))
                    write.save_products(filename, products)
                else:
                        print("Invalid quantity input. Must be a number.")

                    break

                if not found:
                    print("Invalid Product ID. Please try again.")

            except ValueError:
                print("Invalid input. Please enter a numeric Product ID.")

            another_restock = input("Do you want to restock another product? (Y/N): ").upper()
            if another_restock != 'Y':
                break

        if restock_invoice:
            write.create_restock_invoice(restock_invoice)
            write.display_restock_invoice()
        else:
            print("\nNo products were restocked, so no restock invoice was created.\n")
```

Read

```
def read_data(filename):
    """
    ----------------------------------------------------------------------
    Reads product data from a file and returns it as a list of dictionaries.
    ----------------------------------------------------------------------


    Parameters:
        filename (str):
            The name of the file containing product data.


    ----------------------------------------------------------------------
    Returns:
        products (list):
            A list of dictionaries, each representing a product with keys such as
            'Product Id', 'Product Name', 'Brand', 'Quantity', 'Price', and 'Country'.
    ----------------------------------------------------------------------
    """


# Create an empty list to store all products read from the file
    products = []
    '''
Open the file to read product data,
read all lines into a list,
then close the file to free resources.
    '''
    file = open(filename, 'r')
    lines = file.readlines()
    file.close()
```

```
for line in lines:
    index = line.split(',')
    if len(index) == 6:
        serialno = index[0]
        name = index[1]
        brand = index[2]
        quantity = int(index[3])
        cost_price = float(index[4])
        origin = index[5].replace("\n", "")

        product = {
            'S.N': serialno,
            'name': name,
            'brand': brand,
            'quantity': quantity,
            'cost_price': cost_price,
            'origin': origin
        }
        products.append(product)
return products
```