



Islington college
(इस्लिङ्टन कलेज)

CS4051NI/CC4059NI Fundamentals of Computing

70% Individual Coursework

2024/25 Spring

Student Name: Aaki Prajapati

London Met ID: 24046683

College ID: NP01CP4A240116

Assignment Due Date: Wednesday, May 14, 2025

Assignment Submission Date: Wednesday, May 14, 2025

Word Count: 11293

Project File Links:

Onedrive Drive Link:	24046683 AakiPrajapati pythonturnitin.docx
----------------------	--

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.

Table of Contents:

1. Introduction:	1
1.1 About the Project:	1
1.2 Aims and Objectives:	2
1.2.1 Aim:	2
1.2.2 Objectives:	2
1.3 Technology Used:	2
2. Discussion and Analysis	6
2.1 Algorithm:	6
2.2 Flowchart	7
2.3 Pseudocode:	12
2.3.1 read.py:	12
2.3.2 write.py:	12
2.3.3 operations.py:	16
2.3.4 main.py:	25
2.4 Data Structures:	28
2.4.1 Primitive Data Types:	28
2.4.1.1 float:	28
2.4.1.2 str:	29
2.4.1.3 bool:	30
2.4.2 Collection Data Types:	30
2.4.2.1 Lists:	30
2.4.2.2 Tuples:	32
2.4.2.3 Dictionary:	33
2.4.2.4 Sets:	34
3. Program:	35
3.1 Implementation of the Program:	35
3.2 Complete process of purchase and sell:	36
3.2.1 Initial .txt file	36
3.2.2 Sale process:	37
3.2.3 Purchase process:	39
3.2 Creation of .txt file	41
3.3 Opening .txt files to show the bill:	42
3.4 Termination of Program:	43

4. Testing:	43
4.1 Test 1.....	43
4.1 Test 2.....	44
4.1.1 Sale:.....	44
4.1.2 Purchase:	45
4.3 Test 3.....	46
4.4 Test 4.....	48
4.5 Test 5.....	52
5. Conclusion:	56
6. Code Appendix:.....	57
6.1 read.py.....	57
6.2 write.py:	57
6.3 operations.py	64
6.4 main.py:	78
Bibliography	83

Table of Figures:

Figure 1: Python Logo (Python, n.d.).....	3
Figure 2: Microsoft Word (Microsoft, n.d.)	4
Figure 3: Notepad (Microsoft, n.d.).....	4
Figure 4: draw.io (Draw.io, n.d.)	5
Figure 5: Flowchart (initial operations)	8
Figure 6: Flowchart (choice 1).....	9
Figure 7: Flowchart (choice 2).....	10
Figure 8: Flowchart (choice 3).....	11
Figure 9: 'int' data type	28
Figure 10: 'float' data type	29
Figure 11: 'string' data type	29
Figure 12: 'bool' data type	30
Figure 13: Lists in Python.....	31
Figure 14: Tuple with error	32
Figure 15: Tuple without error	32
Figure 16: Dictionary	33
Figure 17: Set.....	34
Figure 18: Initial .txt file	36
Figure 19: Initial display.....	37
Figure 20: Choosing option 1 and Selling item 1	37
Figure 21: Selling item 2 with updated stock	38
Figure 22: Bill generation- sale.....	38
Figure 23: WeCare.txt file after sale	39

Figure 24: Choosing choice 2 and purchasing item 1	39
Figure 25: Purchasing item 3	40
Figure 26: Bill generation- purchase	40
Figure 27: WeCare.txt after purchase	41
Figure 28: Creation of .txt files	41
Figure 29: Creation of unique .txt file after sale	41
Figure 30: Creation of .txt file after purchase	42
Figure 31: Bill generation- sale in unique file.....	42
Figure 32: Bill generation- purchase in unique file	42
Figure 33: Choosing choice 3.....	43
Figure 34: Implementation of try and except block.....	44
Figure 35: Invalid input message displayed for negative and non-existent values (sale)	44
Figure 36: Error message displayed for negative and non-existent values (purchase) .	45
Figure 37: Purchasing item 1	46
Figure 38: Purchasing item 2 and 3.....	47
Figure 39: Bill Generation in shell	47
Figure 40: Purchased product details in .txt file.....	48
Figure 41: Selling item 1.....	49
Figure 42: Selling item 2.....	50
Figure 43: Bill generation in shell	51
Figure 44: Sale product details in .txt file.....	51
Figure 45: Initial .txt file	52
Figure 46: Selling item 1.....	53
Figure 47: .txt file after sale	53
Figure 48: Decreased quantity after sale.....	54
Figure 49: Purchasing item 2	54
Figure 50: Increased quantity after purchase	55
Figure 51: .txt file after purchase	55

Table of Tables:

Table 1: To check try and except block.....	43
Table 2: To check negative and non-existent value as input (sale).....	44
Table 3: To check negative and non-existent value as input (purchase).....	45
Table 4: To show file generation of multiple product purchase	46
Table 5: To show file generation of multiple product sale.....	48
Table 6: To update the stock of products	52

24046683 Aaki Prajapati focfinal turnitin.docx

 Islington College, Nepal

Document Details

Submission ID
trn:oid::3618:95730797

Submission Date
May 14, 2025, 2:48 AM GMT+5:45

Download Date
May 14, 2025, 2:51 AM GMT+5:45

File Name
24046683 Aaki Prajapati focfinal turnitin.docx

File Size
45.5 KB

52 Pages
7,206 Words
38,475 Characters



Page 1 of 57 - Cover Page

Submission ID trn:oid::3618:95730797







Page 2 of 57 - Integrity Overview

Submission ID trn:oid::3618:95730797




8% Overall Similarity

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

Match Groups

-  **66 Not Cited or Quoted 8%**
Matches with neither in-text citation nor quotation marks
-  **2 Missing Quotations 0%**
Matches that are still very similar to source material
-  **1 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 2%  Internet sources
- 0%  Publications
- 8%  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.

1. Introduction:

1.1 About the Project:

The project to develop a Skin Care Product Sale System with the name WeCare was assigned to us by the Fundamentals of Computing faculty with the purpose of gaining more information about the programming language: Python, a well-known and an important sector in the IT field. WeCare is a local vendor who sells beauty and skin care products. It has to keep a track of all the items available in the store. In addition to that, to attract customers, WeCare has provided a “buy three get one free” policy as well. In this coursework, by dividing the program into modules, successful implementation of all required operations were done.

A .txt file with the name WeCare contained all initial product information. The program was divided into 4 modules: main, read, write and operations. All modules were saved with the .py extension. In the main module, all of the functions made across the 3 other modules were imported, and called wherever required. The read module contained function which read all details from the .txt file containing product information. Similarly, the write module contained all set of codes that were to be displayed to the user, or written to a unique file. Finally, the operations module contained the main logic. Operations such as taking input, calculating price, adding details to a list, etc. were performed in this module. The text file was updated with each operation performed. The concept of 2D lists was used to read data from the .txt file, access and update the values.

As mentioned beforehand, the concept of file handling was applied. The text file stored in the same file location as the python files was opened using the open() function. The read function takes filename and mode in which we want the file to be dealt with as parameters. The read mode, denoted by “r” was used to display the required stock. Then, the write mode denoted by “w” was used to update the stock, and write product details in the text files, whenever a customer bought something, or when we restock the products.

The concept of lists were majorly used in this coursework. Lists are used to store items in an ordered sequence, which can be accessed through index. The data from the text file was converted to a 2D list, which allowed us to update stock, access prices, quantity, names of products and make use of them to carry out all required operations.

In short, making use of 2D lists, file handling, and modularity, this coursework was implemented successfully

1.2 Aims and Objectives:

1.2.1 Aim:

- The aim of this project is to develop and maintain a wholesale skin and beauty product sale system using Python, which actively maintains the goods available, updates them, maintains the invoices of each customer and follows the “buy three get one free” offer. Furthermore, it is to teach us to learn the concepts, and apply it in real life scenarios as well.

1.2.2 Objectives:

- To efficiently maintain the product stock and display the details of the items available.
- To produce correct invoices for each customer, with accurate number of items, their respective prices, and free products if 3 items are bought, having unique file name.
- To update the stock each time a customer buys something, including the free items.
- To update the stock if restock is done.
- To generate a unique file for bill after restock.
- To promote user interaction.

1.3 Technology Used:

i. Python:

Python is a high-level, interpreted, flexible and dynamic programming language. Provided with a complete standard library, Python emphasizes upon the reusability and productivity of the code. It also focuses upon the modularity of the code, making programming easier and efficient. This programming language has been gaining more popularity day by day, due to its easy syntax, and multiple functionality in various sectors: AI, Web development, general purpose programs, etc. It supports the procedural, functional and object oriented programming paradigms. Being an interpreted language, flexibility of code increases, and debugging becomes easier. All

in all, its portability, large community and readability has allowed developers to become more fascinated towards Python.

How it was used in the coursework

- IDLE version 3.13 was used to compute the .py files in this project. A full product catalog, stock management system using modular programming was built using Python. Using simple calculations, our everyday processes of buying/ selling products was done, along with successful stock update and bill calculation. Through this platform, the execution of the WeCare Wholesale system became more simple, readable and easy to debug.



Figure 1: Python Logo (Python, n.d.)

ii. Microsoft Word:

Microsoft Word is a software designed to allow users to create, store, use, share, edit and delete the files. It has been widely known all over the world for its simple and flexible use in creation of any kind of file, along with additional features such as adding images, inserting tables, figures and much more.

How it was used in the coursework:

- In this coursework, MS-Word was used to create the report. After the completion of the requirements of this coursework, report to show evidences of the working of the program through testing, images, and explanations were done in this platform providing an efficient way to create the report.



Figure 2: Microsoft Word (Microsoft, n.d.)

iii. Notepad:

Notepad is a built in application for Windows, which is a simple text editor to create, edit, format and delete files. The files are stored in .txt format which is very useful for a number of programming languages to retrieve, store and update the data stored. Being a simple and easy to use application, its popularity and use has increased worldwide.

How it was used in the coursework:

- In this coursework, notepad was used to generate a unique file for each purchase and sell. Each of the files created had a bill generated, with the product name, quantity, price, and other calculated outputs. It allowed us to have a platform where we could see how many products were bought/ sold, along with the total price, making the program flow even more effective.



Figure 3: Notepad (Microsoft, n.d.)

iv. Draw.io:

Draw.io is an online tool used for the making various different types of diagrams, such as flowcharts, ER diagrams, mind maps, etc. It allows the user to create a visual representation of their implementation of programs, allowing clearer implementation in real life. Its easy interface has also attracted many users. Since users do not need to have any special training to use draw.io, it has become a more preferred platform to build visual concepts.

How it was used in the coursework:

- In the coursework, draw.io was used to create flowchart of the program flow. Making use of its easy drag and drop functionality, the process of creating the flowchart to visualize the flow of program became more simple and efficient.



Figure 4: draw.io (Draw.io, n.d.)

- Concepts:

The concepts of lists, 2D lists, strings, loops, etc. were used to precisely showcase the available items in the store, to generate bills, update stock, provide free items and much more.

2. Discussion and Analysis

2.1 Algorithm:

An algorithm is a set of steps that defines the flow of a program. It does not have a standard to be followed, however these set of steps must perform the operations required. Since it is not bound to any particular programming language, the use of these set of steps allows a clearer vision and understanding of what program has been developed. Hence, this process of breaking down any program into a set of steps not only allows an efficient and easy to understand, but also makes it easier to debug or find errors with ease.

Algorithm of the Program:

STEP 1: Display the shop information, and product information from .txt file

STEP 2: Display choices to the user, and allow them to enter any 1 choice

STEP 2.1: IF choice entered by user is 1 then, go to Step 3

STEP 2.2: ELSE IF choice entered by user is 2 then, go to Step 9

STEP 2.3: ELSE IF choice entered by user is 3 then, go to Step 13

STEP 2.4: ELSE go to Step 2

STEP 3: Ask user's name

STEP 4: Ask user's phone number

STEP 4.1: IF length of phone number is not equal to 10 repeat step 4

STEP 4.2: ELSE go to step 5

STEP 5: Ask ID of product to be purchased by customer

STEP 5.1: IF the ID of product is invalid, go to STEP 5

STEP 5.2: ELSE go to STEP 6

STEP 6: Ask quantity of product to be purchased by customer

STEP 6.1: IF the quantity of product is invalid, go to STEP 6

STEP 6.2: ELSE go to STEP 7

STEP 7: Ask the user if they want to purchase more

STEP 7.1: If the user enters "yes" or "y" go to STEP 5

STEP 7.2: ELSE go to STEP 8

STEP 8: Display the bill with total price and go to STEP 2

STEP 9: Ask ID of product to be restocked by the user

STEP 9.1: IF the ID of product is invalid, go to STEP 9

STEP 9.2: ELSE go to STEP 10

STEP 10: Ask quantity of product to be restocked by the user

STEP 10.1: IF the quantity of product is invalid, go to STEP 10

STEP 10.2: ELSE go to STEP 11

STEP 11: Ask the user if they want to restock more items

STEP 11.1: If the user enters “yes” or “y” go to STEP 9

STEP 11.2: ELSE go to STEP 12

STEP 12: Display the bill with total price and go to STEP 2

STEP 13: Exit the system.

2.2 Flowchart

A flowchart is a visual/ graphical representation of an algorithm, making use of various shapes to denote different operations. The use of flowchart allows developers to have an idea about the program flow, hence creating a vision of how the required program could possibly be developed. Hence it could also be considered as a blueprint, acting as a guide for program development.

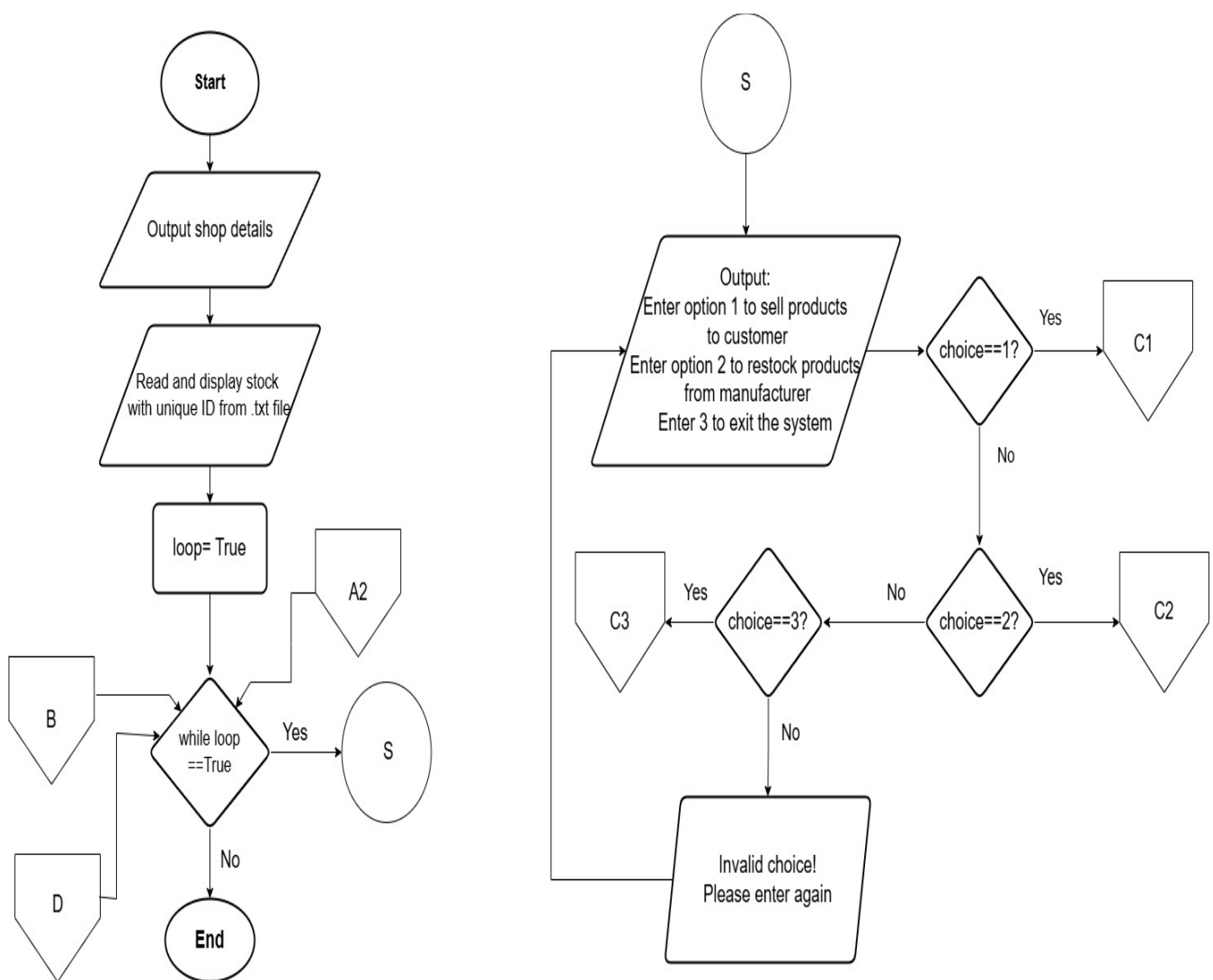


Figure 5: Flowchart (initial operations)

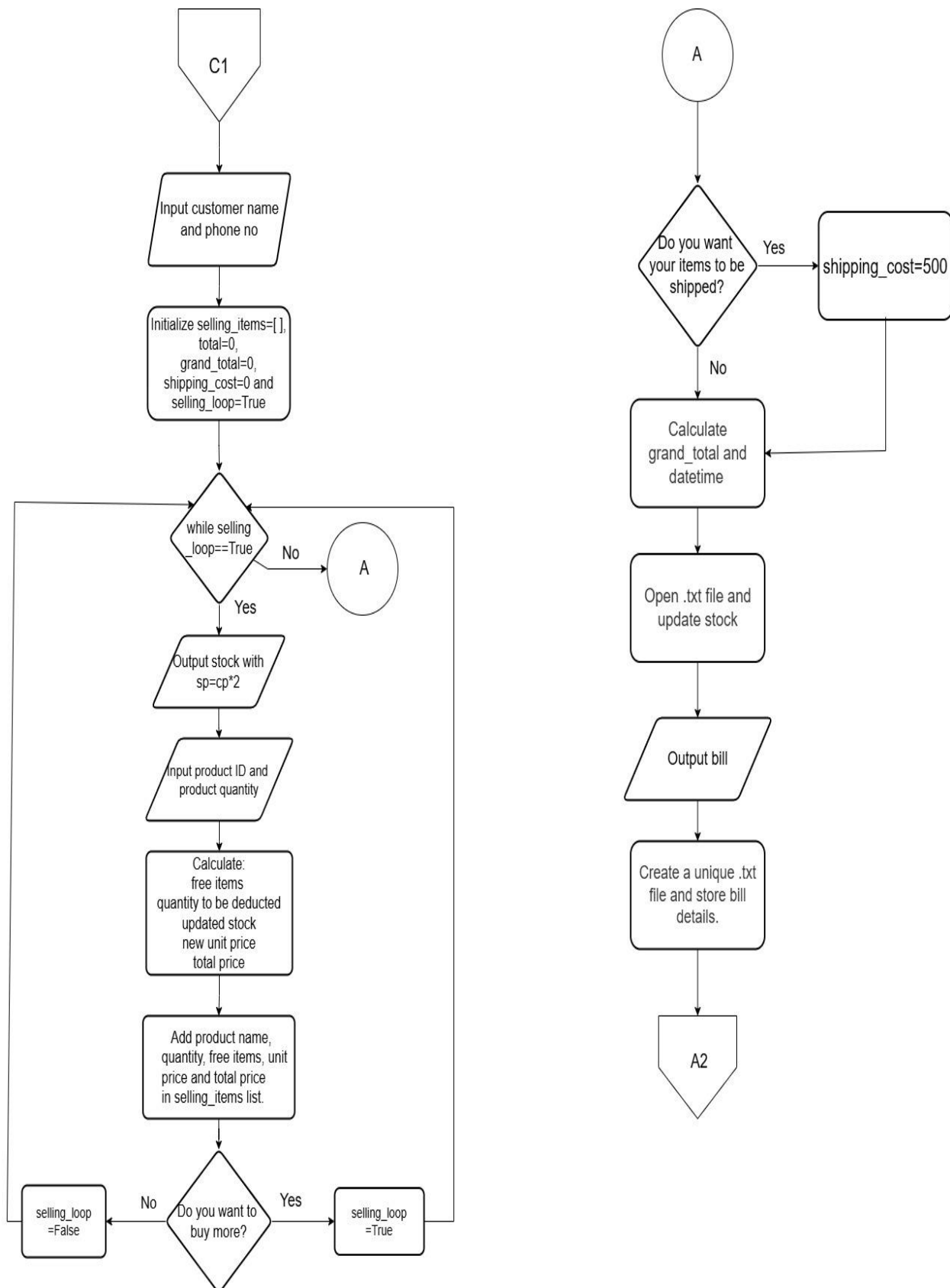


Figure 6: Flowchart (choice 1)

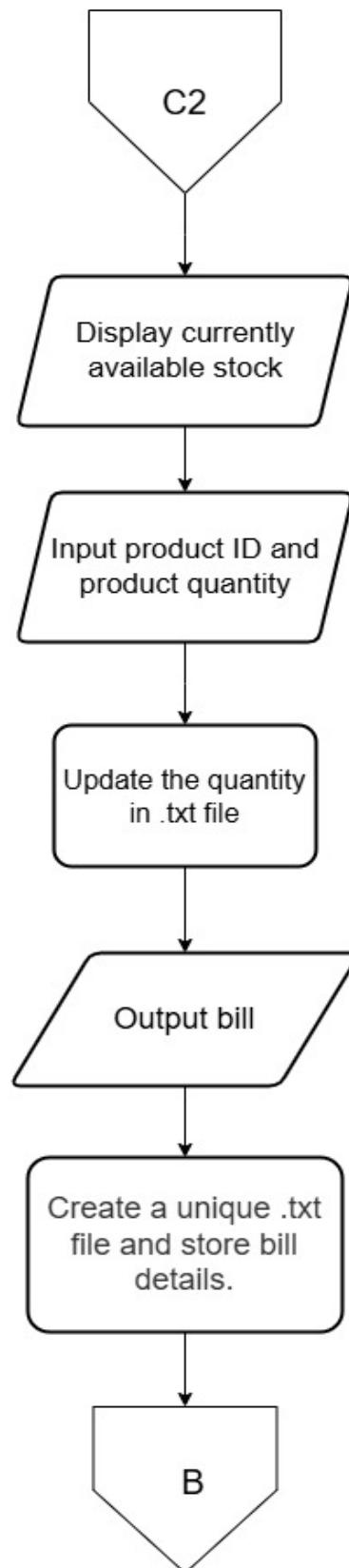


Figure 7: Flowchart (choice 2)

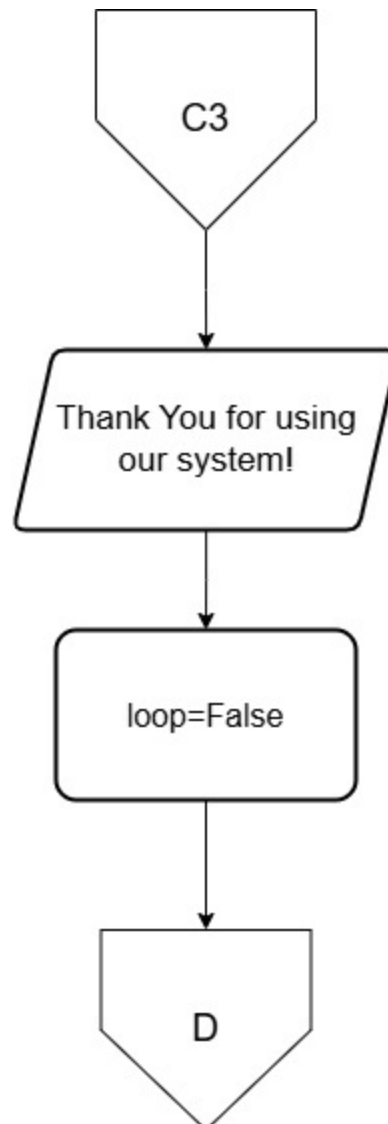


Figure 8: Flowchart (choice 3)

2.3 Pseudocode:

A pseudocode is a detailed but a simple description that explains what a program does. It has no specific set of rules, but is a way to describe programs in human language in such a way that other programmers besides the developer can understand and execute the program.

2.3.1 read.py:

```
DEFINE FUNCTION read_from_file()
    OPEN file "WeCare.txt" in read mode
    INITIALIZE nl as an empty list
    READ all lines from the file and store it in the list lines
    FOR each in lines
        REPLACE newline character by "" ,split the elements on the basis of "," and
            store it in list l
        ADD the new list l to nl
    END FOR
    CLOSE file
    RETURN list nl
END FUNCTION read_from_file()
```

2.3.2 write.py:

```
DEFINE FUNCTION initial_shop_details()
    DISPLAY shop details
    DISPLAY welcome message
END FUNCTION initial_shop_details()
```

```
DEFINE FUNCTION user_display()
    DISPLAY "Dear Admin, please choose any one option from the choices below to
        carry out the operations."
    DISPLAY "Please enter 1 to sell the products to the customer."
    DISPLAY "Please enter 2 to purchase products from the manufacturer."
    DISPLAY "Please enter 3 to exit from the system."
```

END FUNCTION user_display()

DEFINE FUNCTION bill_info()

DISPLAY a message to user to inform them about details required for bill
 generation

END FUNCTION bill_info()

DEFINE FUNCTION display_free_items(name,free_products)

DISPLAY a message containing number of free products obtained by the user

END FUNCTION display_free_items(name,free_products)

DEFINE FUNCTION update_after_purchase (nl)

OPEN file "WeCare.txt" in write mode

FOR i in range from 0 till length of nl minus 1

FOR j in range from 0 till length of nl[i] minus 1

WRITE nl[i][j] in file

IF j is not equal to length of (nl[i]) minus1

WRITE "," in file

END IF

END FOR

END FOR

CLOSE file

END FUNCTION update_after_purchase (nl)

DEFINE FUNCTION purchase_bill_onscreen(name, phone_no, date_and_time_tdy)

DISPLAY shop details, customer name, phone number and current date and time

DISPLAY headings including Item, Quantity, Free Items, Unit Price, Total

END FUNCTION purchase_bill_onscreen(name, phone_no, date_and_time_tdy)

DEFINE FUNCTION restock_bill_onscreen(date_and_time_tdy)

DISPLAY shop details, customer name, phone number and current date and time

```
DISPLAY the headings including Item, Quantity, Unit Price, Total
END FUNCTION restock_bill_onscreen(date_and_time_tdy)

DEFINE FUNCTION unique_bill(name, phone_no, date_and_time_tdy, shipping_cost,
                             grand_total, selling_items)
    OPEN file name+str(phone_no)+".txt" in write mode
    WRITE shop details, customer name, phone number and current date and time
    WRITE headings including Item, Quantity, Unit Price, Total
    FOR a in selling_items
        WRITE a[0]+"\\t " +str(a[1])+ "\\t\\t " +str(a[2])+ "\\t\\t" +"$"+str(a[3])+ "\\t\\t"
            +"$"+ str(a[4])+"\\n"
    END FOR
    IF shipping_cost is greater than 0 THEN
        WRITE shipping_cost
    END IF
    WRITE grand_total
    CLOSE file
END FUNCTION unique_bill(name, phone_no, date_and_time_tdy, shipping_cost,
                           grand_total, selling_items)

DEFINE FUNCTION file_after_restock(nl)
    OPEN file "WeCare.txt" in write mode
    FOR i in range from 0 till length of nl minus 1
        WRITE nl[i][0]+", "+nl[i][1]+", "+nl[i][2]+", "+nl[i][3]+", "+nl[i][4]
        WRITE "\\n"
    END FOR
    CLOSE file
END FUNCTION file_after_restock(nl)

DEFINE FUNCTION bill_after_restock(unique_num, total_of_all, purchase,
                                     date_and_time_tdy)
```

```
OPEN file unique_num+".txt" in write mode
WRITE shop details, customer name, phone number and current date and time
WRITE headings including Item, Quantity, Unit Price, Total
FOR a in purchase
    WRITE a[0]+"\\t " +str(a[1])+ "\\t\\t" +"$" +str(a[2])+ "\\t\\t" +"$" + str(a[3])+"\\n"
END FOR
WRITE grand_total
CLOSE file
END FUNCTION bill_after_restock(unique_num, total_of_all, purchase,
                                date_and_time_tdy)

DEFINE FUNCTION option_3()
    DISPLAY "Thank You for choosing us. Have a great day!"
END FUNCTION option_3()

DEFINE FUNCTION invalid_choice(choices)
    DISPLAY "Sorry! The choice",choices,"does not exist. Please enter the correct
            choice."
END FUNCTION invalid_choice(choices)
```

2.3.3 operations.py:

IMPORT datetime module

DEFINE FUNCTION display_stock(nl)

INITIALIZE product_id AS 1

DISPLAY heading containing Id, Name, Brand, Qty, Price and Origin

FOR i in range from 0 till length of nl minus 1

DISPLAY product_id

FOR j in range from 0 till length of nl[i] minus -1

DISPLAY nl[i][j]

END FOR

INCREASE product_id by 1

END FOR

END FUNCTION display_stock(nl)

DEFINE FUNCTION input_choices()

DECLARE choice_loop as False

WHILE choice_loop equals to False

TRY

ASK the user to enter their choice in the form of integer

ASSIGN choice_loop as True

CATCH exception

DISPLAY error message

ASSIGN choice_loop as False

END TRY

END WHILE

RETURN user's choice

END FUNCTION input_choices()

DEFINE FUNCTION enter_user_details ()

ASK the name of the customer

```
ASK the phone number of the customer
WHILE length of phone number is LESS THAN 10 OR length of phone number is
GREATER THAN 10
    DISPLAY message phone number must be 10 digits
    ASK the phone number of the customer
END WHILE
RETURN name, phone number of customer
END FUNCTION enter_user_details ()

DEFINE FUNCTION initialize_variable()
    INITIALIZE selling_items as empty list, total as 0, grand_total as 0, shipping_cost
as 0, selling_loop as True
    RETURN the initialized variables
END FUNCTION initialize_variable()

DEFINE FUNCTION buying_product_id(nl)
    INITIALIZE buying_loop as False
    WHILE buying_loop IS EQUAL TO False
        TRY
            ASK the ID of product the user wants to buy in the form of integer
            CALL FUNCTION buying_id_validation (buying_id, nl)
            ASSIGN buying_loop as True
        CATCH exception
            DISPLAY Error Message
            ASSIGN buying_loop as False
        END TRY
    END WHILE
    RETURN valid product ID entered by user
END FUNCTION buying_product_id(nl)
DEFINE FUNCTION buying_product_qty(nl)
    INITIALIZE purchase_quantity_loop as False
```

```
WHILE purchase_quantity_loop is equal to False
    TRY
        ASK the quantity of product that the user wants to buy in the form of
        integer
        ASSIGN purchase_quantity_loop as True
    CATCH exception
        DISPLAY Error Message
        ASSIGN purchase_quantity_loop as False
    END TRY
END WHILE
RETURN the product quantity entered by user
END FUNCTION buying_product_qty(nl)
```

```
DEFINE FUNCTION purchase_product(product_quantity,nl,buying_id)
    CALCULATE free products by dividing product quantity by 3 and rounding it off to
    a whole number
    CALCULATE the quantity to be deducted from stock by adding product quantity
    and number of free items
    SET the value of available quantity of selected item to the value at second index of
    the (buying_id -1)th item in nl
    WHILE the product quantity is less than or equal to 0 OR quantity to be deducted
    from stock is greater than the integer value of available quantity of selected item
        DISPLAY message "Quantity is not available"
        ASK the user to enter the product quantity again
        CALCULATE free products by dividing product quantity by 3 and rounding
        it off to a whole number
        CALCULATE the quantity to be deducted from stock by adding product
        quantity and number of free items
    END WHILE
    RETURN the number of free items, quantity to be deducted from stock, and the
    validated product quantity
```

END FUNCTION purchase_product(product_quantity,nl,buying_id)

DEFINE FUNCTION buy_more()

ASK the user if they want to purchase more and convert the answer to lower case

IF the user's answer is "y" or "yes" **THEN**

ASSIGN selling_loop as True

ELSE

ASSIGN selling_loop as False

END IF

RETURN the boolean value of selling_loop

END FUNCTION buy_more()

DEFINE FUNCTION stock_display_choice1(nl)

DISPLAY the headings that contains ID, Name, Brand, Qty, Price, Origin

INITIALIZE product_id as 1

FOR i in range from 0 to length of nl minus 1

DISPLAY product ID

FOR j in range from 0 to length of nl[i] minus 1

IF j is equal to 3 **THEN**

DISPLAY CP as initial CP*2, i.e. $nl[i][j]*2$

ELSE

DISPLAY $nl[i][j]$

END IF

END FOR

INCREASE product_id by 1

END FOR

END FUNCTION stock_display_choice1(nl)

DEFINE FUNCTION buying_id_validation(buying_id, nl)

WHILE ID of product that the user wants to buy is less than or equal to 0 OR ID of product that the user wants to buy is greater than the length of nl

DISPLAY Error Message

ASK the user to enter the product ID again

END WHILE

RETURN the validated ID entered by the user

END FUNCTION buying_id_validation(buying_id, nl)

DEFINE FUNCTION update_stock_after_purchase(buying_id, product_quantity,
free_products, quantity_to_deduct, selling_items, total, nl)

REDUCE the previous stock quantity with total quantity to be deducted, and update
the stock quantity

GET the required product name from 2D list nl

GET the required unit price from 2D list nl and multiply it by 2

CALCULATE the total price without free items as unit price* product quantity

ADD product name, quantity, number of free products, unit price and total price of
the particular item in the selling_items list

CALCULATE total price of all items as total of all items+ total price of each item

RETURN selling_items list and total price of all items

END FUNCTION update_stock_after_purchase(buying_id, product_quantity,
free_products, quantity_to_deduct, selling_items, total, nl)

DEFINE FUNCTION shipping(shipping_cost)

ASK the user if they want their products to be shipped and convert their answer to
lower case

IF the user's answer is "y" or "yes" **THEN**

ASSIGN shipping_cost=500

END IF

RETURN the value of shipping_cost

END FUNCTION shipping(shipping_cost)

DEFINE FUNCTION final_total(grand_total, total, shipping_cost)

CALCULATE grand_total as sum of total price of all items and shipping cost\

RETURN grand_total

END FUNCTION final_total(grand_total, total, shipping_cost)

DEFINE FUNCTION date_time_tdy()

SET date_and_time_tdy as obtained from datetime module's datetime class's
 now() function

RETURN date_and_time_tdy

END FUNCTION date_time_tdy()

DEFINE FUNCTION bill_items(selling_items, shipping_cost, grand_total)

FOR a in selling_items list

DISPLAY corresponding values present in the list using indexes in the form
 of string

END FOR

IF shipping_cost is greater than 0 **THEN**

DISPLAY shipping_cost

END IF

DISPLAY grand_total price which is the total price of all items and shipping cost

END FUNCTION bill_items(selling_items, shipping_cost, grand_total)

DEFINE FUNCTION variables()

INITIALIZE purchase list as empty, total_of_all as 0 and purchase_loop as True

RETURN the initialized variables

END FUNCTION variables()

DEFINE FUNCTION restock_id(nl)

INITIALIZE restock_loop as False

WHILE restock_loop is equal to False

TRY

ASK user to enter the ID of the product they want to restock in the
 form of integer

CALL FUNCTION restock_id_validation(restock_product_id,nl)

```
        ASSIGN restock_loop as True
    CATCH exception
        DISPLAY Error Message
        ASSIGN restock_loop as False
    END TRY
END WHILE
RETURN the validated product ID
END FUNCTION restock_id(nl)

DEFINE FUNCTION restock_qty(nl)
    INITIALIZE restock_quantity_loop as False
    WHILE restock_quantity_loop is equal to False
        TRY
            ASK quantity of product that the user wants to restock in the form of
            integer
            CALL FUNCTION restock_quantity_validation(restock_product_qty)
            ASSIGN restock_quantity_loop as True
        CATCH exception
            DISPLAY Error Message
            ASSIGN restock_quantity_loop as False
        END TRY
    END WHILE
    RETURN the validated product quantity
END FUNCTION restock_qty(nl)

DEFINE FUNCTION restock_again()
    DISPLAY Successful message of stock update
    ASK the user if they want to restock more, and convert their answer into lower case
    IF the user's answer is "y" OR "yes" THEN
        ASSIGN purchase_loop as True
    ELSE
```

```
        ASSIGN purchase_loop as False
    END IF
    RETURN the boolean value of purchase_loop
END FUNCTION restock_again()

DEFINE FUNCTION restock_bill_items(total_of_all, purchase)
    FOR a in purchase list
        DISPLAY corresponding values present in the list using indexes in the form
        of string
    END FOR
    DISPLAY the grand total as string value of total_of_all
END FUNCTION restock_bill_items(total_of_all, purchase)

DEFINE FUNCTION unique_date_time()
    ASSIGN year, month, day, hour, minute, second from datetime module's datetime
    class's now() function in the form of string
    CONCATENATE year, month, day, hour, minute, second and store it in
    unique_num
    RETURN the value of unique_num
END FUNCTION unique_date_time()

DEFINE FUNCTION stock_display_choice2(nl)
    DECLARE product_id as 1
    DISPLAY the suitable messages and headings including ID, Name, Brand, Qty,
    Price, Quantity
    FOR i in range from 0 to length of nl minus 1
        DISPLAY product_id
        FOR j in range from 0 to length of nl[i]
            DISPLAY nl[i] [j]
        END FOR
    INCREASE product_id by 1
```

```
END FOR
END FUNCTION stock_display_choice2(nl)

DECLARE FUNCTION restock_id_validation(restock_product_id,nl)
    WHILE restock_product_id is less than or equal to 0 or restock_product_id is
    greater than the length of nl
        DISPLAY Error Message
        ASK the user to enter the ID of product they want to restock again
    END WHILE
    RETURN the validated product ID
END FUNCTION restock_id_validation(restock_product_id,nl)

DECLARE FUNCTION restock_quantity_validation(restock_product_qty)
    WHILE restock_product_qty is less than or equal to 0
        DISPLAY Error Message
        ASK the user to enter the quantity of product they want to restock again
    END WHILE
    RETURN the validated quantity of product that the user wants to restock
END FUNCTION restock_quantity_validation(restock_product_qty)

DECLARE FUNCTION stock_after_restock(restock_product_id, restock_product_qty,nl)
    UPDATE the quantity in stock of the 2D list nl by adding the previous quantity with
    the restock_product_qty as provided by the user
    RETURN the updated list nl
END FUNCTION stock_after_restock(restock_product_id, restock_product_qty,nl)
DECLARE FUNCTION restock_bill(nl, restock_product_id, restock_product_qty,
purchase, total_of_all)
    GET the required product name from 2D list nl
    GET the required unit price from 2D list nl
    CALCULATE the total price by multiplying restock_product_qty and
    restock_unit_price
```

ADD restock product name, qty, unit price and total price to the purchase list

CALCULATE total_of_all which is the total price including all items

RETURN the list purchase and total_of_all items

END FUNCTION restock_bill(nl, restock_product_id, restock_product_qty, purchase, total_of_all)

2.3.4 main.py:

IMPORT datetime module

IMPORT read_from_file from read_AakiPrajapati

IMPORT import unique_date_time, restock_bill_items, restock_id, restock_again, restock_qty, variables, bill_items, date_time_tdy, final_total, buying_product_qty, buy_more, buying_product_id, buying_product_id, enter_user_details, initialize_variable, display_stock, input_choices, stock_display_choice1, buying_id_validation, purchase_product, update_stock_after_purchase, shipping, stock_display_choice2, restock_id_validation, restock_quantity_validation, stock_after_restock, restock_bill from operations_AakiPrajapati

IMPORT invalid_choice, option_3, restock_bill_onscreen, purchase_bill_onscreen, display_free_items, bill_info, initial_shop_details, user_display, update_after_purchase, unique_bill, file_after_restock, bill_after_restock from write_AakiPrajapati

CALL FUNCTION initial_shop_details()

CALL FUNCTION read_from_file()

CALL FUNCTION display_stock(nl)

INITIALIZE loop as True

WHILE loop is equal to True

CALL FUNCTION user_display()

CALL FUNCTION input_choices()

IF choices is equal to 1 **THEN**

CALL FUNCTION bill_info()

CALL FUNCTION enter_user_details()

```
CALL FUNCTION initialize_variable()
WHILE selling_loop is equal to True
    CALL FUNCTION stock_display_choice1(nl)
    CALL FUNCTION buying_product_id(nl)
    CALL FUNCTION buying_product_qty(nl)
    CALL FUNCTION purchase_product(product_quantity,nl,buying_id)
    CALL FUNCTION display_free_items(name,free_products)
    CALL FUNCTION update_stock_after_purchase(buying_id,
        product_quantity, free_products,
        quantity_to_deduct,selling_items, total, nl)
    CALL FUNCTION buy_more()
END WHILE
CALL FUNCTION shipping(shipping_cost)
CALL FUNCTION final_total(grand_total, total, shipping_cost)
CALL FUNCTION date_time_tdy()
CALL FUNCTION update_after_purchase(nl)
CALL FUNCTION purchase_bill_onscreen(name, phone_no,
    date_and_time_tdy)
CALL FUNCTION bill_items(selling_items, shipping_cost, grand_total)
CALL FUNCTION unique_bill(name, phone_no, date_and_time_tdy,
    shipping_cost, grand_total, selling_items)
```

```
ELSE IF choices is equal to 2
    CALL FUNCTION variables()
    WHILE purchase_loop is equal to True
        CALL FUNCTION stock_display_choice2(nl)
        CALL FUNCTION restock_id(nl)
        CALL FUNCTION restock_qty(nl)
        CALL FUNCTION stock_after_restock(restock_product_id,
            restock_product_qty,nl)
        CALL FUNCTION file_after_restock(nl)
```

```
        CALL FUNCTION restock_bill(nl, restock_product_id,  
                                   restock_product_qty, purchase, total_of_all)  
        CALL FUNCTION restock_again()  
    END WHILE  
    CALL FUNCTION date_time_tdy()  
    CALL FUNCTION restock_bill_onscreen(date_and_time_tdy)  
    CALL FUNCTION restock_bill_items(total_of_all, purchase)  
    CALL FUNCTION unique_date_time()  
    CALL FUNCTION bill_after_restock(unique_num, total_of_all, purchase,  
                                     date_and_time_tdy)  
    ELSE IF choices is equal to 3  
        CALL FUNCTION option_3()  
        ASSIGN loop as False  
    ELSE  
        CALL FUNCTION invalid_choice(choices)  
    END IF  
END WHILE
```


2.4 Data Structures:

2.4.1 Primitive Data Types:

Primitive Data Types refer to the immutable, and the most basic, yet the most important data types in Python. They are used to represent integer, float, Boolean and string values.

2.4.1.1 int:

The 'int' data type is used to represent integers. It consists of positive and negative whole numbers (numbers without fractions or decimals). They are immutable by nature.

Example: Numbers such as 5, -7, 8, etc. are of int data type.

Use: Variables/ values of int data type are used for mathematical calculations, in loops as counters and also to access elements from lists, tuples or strings as index.

The data type 'int' can be implemented in the following ways:

```
#Taking Product ID from the customer
buying_id=int(input("Please enter the ID of the product you want to buy: "))
```

Figure 9: 'int' data type

In the above screenshot, the int data type has been used before the input statement. In python, whenever an input is taken from a user, it is taken in the form of a string. Now, the required input is an integer, and we might have to perform arithmetical or conditional operations on the inputted value. So, for this the input from user is converted into an integer denoted by the keyword 'int'. The entire process is called type conversion.

2.4.1.2 float:

The 'float' data type is used to represent real numbers. It consists of numbers specified by decimal point. They are immutable by nature.

Example: Real numbers such as 3.1, -7.0, 9.8, etc. are of float data type.

Use: Variables/ values of float data type are used for precise mathematical calculations, since they provide answers in decimal format.

In a python program, the use of 'float' data type can be done in the following ways:

```

a=3.0
b=2
print(a*b)

= RESTART: C:/Users/DELL/Desktop/uni/fundamentals of computing/python programs/r
eportlaieg.py
6.0

```

Figure 10: 'float' data type

In the above screenshot, variable 'a' is declared as 3.0 which is of float data type. In python, we don't have to declare the data type for the variables, it is understood by default. The float data type is used to denote the exact number, since it consists of decimals, in comparison to int data type. Now, the output is also in the form of float because the variable 'a' is of float data type, and python converts the multiplication of int and float into float to avoid loss of precision.

2.4.1.3 str:

The 'str' or string data type is used to represent characters, words or sentences. There is no 'char' data type in python, so a character is referred to as a string of unit length. They can be accessed through index. Strings are enclosed within single quotation or double quotation marks. They are immutable by nature. They are similar to lists, and can be accessed by index. Functions such as len(), split(), replace(), count(), lower(), upper() etc. are applicable in strings.

Example: "hello world", 'hello world'

Use: Strings are used to represent any data that are not integers.

The use of 'string' data type can be done in the following way:

```

a=input("enter a number")
b=input("enter second number")
print(a+b)

=== RESTART: C:/Users/DELL/Desktop/uni/fundamentals of computing/python programs/reportlaieg.py ===
enter a number5
enter second number10
510

```

Figure 11: 'string' data type

In the above screenshot, a and b are taken as input from the user, where the instruction asks to enter a number. This number is stored in the form of string as "5" and "10" since

the type conversion has not been done. Now, since these numbers are string and not integers, the '+' operator concatenates the two strings instead of addition. Therefore, the output of "5"+"10"="510" and not 15.

2.4.1.4 bool:

The 'bool' data type is used to represent Boolean values. They have two values, 'True' or 'False'. Especially with comparison, it returns the value 'True' if the result is correct, and 'False' if not. The use of bool data type is also done in loops. If the required condition of loop is satisfied, it returns the Boolean value True and proceeds with the remaining logic within the loop.

Example: `print (2>4)` #output: False

Use: Boolean data type is used to return the value True or False after comparison.

The 'bool' data type could be used in the following ways:

```
#Checking if the product ID is valid
while buying_id<=0 or buying_id>len(nl):      #Checking if the admin has entered an id less than 0 o
    print("Sorry! The ID you have entered is invalid. Please enter a valid ID!")
```

Figure 12: 'bool' data type

In the above screenshot, a while loop is being used. The <= and > sign compares the values, and evaluates the result to return in the form of a Boolean value, either True or False. If the given condition gives the Boolean value True, the while loop executes the steps below. If the condition is not satisfied, and the Boolean value False is returned by the expression, then the loop is not executed.

2.4.2 Collection Data Types:

Unlike primitive data types, collection data type can be used to store a number of elements within 1 variable. They may or may not be mutable. They can also store elements of different data type, may it be of primitive or collection data type.

2.4.2.1 Lists:

List is an ordered collection of data where items of different types can be stored together. A square bracket [] is used to enclose the items within a list. The concept of positive and negative indexing is used to access the items. The positive indexing starts from 0 to

`len(list)-1` from the left side of the list. Here the `len()` function is used to find the length of a particular list. Similarly, the negative indexing represent positions from the end of an array, i.e -1 represents the last item, -2 represents the second last item, and so on. Moreover, lists are mutable.

Example: `a= [1, "hello world", 3.14]`

`a` is a list containing 3 items, all of different data types. Its length is 3, and the index starts from 0 to 2.

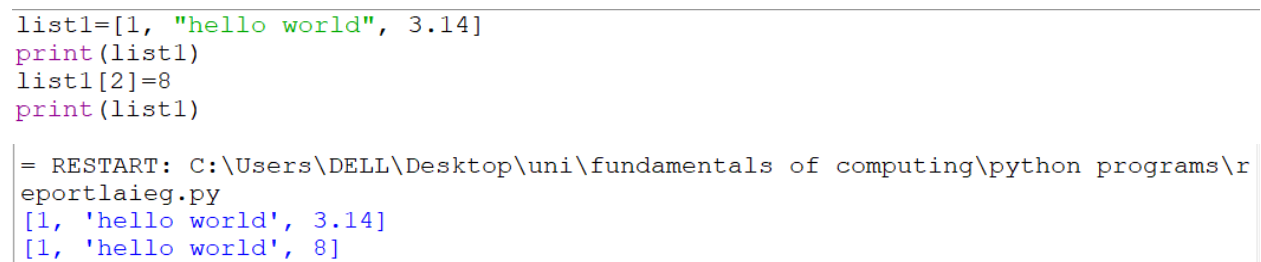
However if a code is written as: `a [2]= 8`

Then, the list changes to: `a=[1, "hello world", 8]`

This proves that lists are mutable.

Use: Lists are used to store huge amount of data at once. They are used to store multiple items within one variable. For instance, to store the marks of 5 subjects of a particular student, a list is used, where their name, and marks of respective subjects are stored in sequential manner.

List can be implemented in the following ways:



```
list1=[1, "hello world", 3.14]
print(list1)
list1[2]=8
print(list1)
```

```
= RESTART: C:\Users\DELL\Desktop\uni\fundamentals of computing\python programs\re
portlaieg.py
[1, 'hello world', 3.14]
[1, 'hello world', 8]
```

Figure 13: Lists in Python

In the above screenshot, the list '`a`' contains 3 elements, each of different data type. The print statement shows the list. As mentioned above, lists are mutable. Therefore, we can add, delete, replace elements into an existing list. Out of the possible operations, in figure 7, I have replaced the value in index 2. Now, when the list is printed again, the new element is stored in the second index. Lists are important to store large number of data, and run the system according to that.

2.4.2.2 Tuples:

Tuples are similar to lists, but immutable in nature. They are represented within parentheses (). Similar to lists, tuples are also an ordered collection of data where data of multiple types can be stored. The concept of positive and negative indexing is applicable here as well for the purpose of accessing the elements within it.

Example: `a= (1, "hello world", 3.4)`

But, if a code is written as: `a[2]=8`, an error message is seen.

This proves that Tuples are immutable, unlike lists.

Use: Tuples are also used to store multiple data within a variable, but the values will remain as constants.

Tuples can be implemented in the following way:

```
tuple1=(1, "hello world", 3.14)
print(tuple1)
tuple1[2]=8
print(tuple1)
```

```
= RESTART: C:\Users\DELL\Desktop\uni\fundamentals of computing\python programs\re
eportlaieg.py
(1, 'hello world', 3.14)
Traceback (most recent call last):
  File "C:\Users\DELL\Desktop\uni\fundamentals of computing\python programs\repo
rtlaieg.py", line 3, in <module>
    tuple1[2]=8
TypeError: 'tuple' object does not support item assignment
```

Figure 14: Tuple with error

Here, an error is shown when I tried to change the element in a particular index. This shows that tuples are immutable; i.e they cannot be changed. Tuples could be used to store constant data. The fixed code is:

```
tuple1=(1, "hello world", 3.14)
print(tuple1)
```

```
= RESTART: C:\Users\DELL\Desktop\uni\fundamentals of computing\python programs\re
eportlaieg.py
(1, 'hello world', 3.14)
```

Figure 15: Tuple without error

Now, the tuple here is just declared and printed to the console. Each element could also be accessed using for loop and indexes. Hence, it doesn't perform operations of adding, replacing or deleting data present in the provided index.

2.4.2.3 Dictionary:

A dictionary is an unordered collection of key-value pairs enclosed within curly brackets {}. The keys are immutable whereas values are mutable and can contain duplicates. The key and values are separated by a colon (:), and each pair is separated by a comma (.). Since it is an unordered collection, indexing cannot be used and hence the exact position of items cannot be determined. However, to gain access to each of the key, value or key-value pairs, for each loop can be used. In the newer versions of Python (older than 3.7) dictionary is an ordered collection.

Example: `d={'id': 1, 'name': 'Aaki'}`

Here, id and name are keys whereas 1 and Aaki are its values respectively.

To access the value of any key, we can write:

```
print (d['key'])
```

Use: In context of large amount of data, when using lists, it is difficult to refer to the index of particular item through a lot of lists. To solve this, the concept of dictionary is used, where instead of having to remember the index, we can access the required items through keys.

Dictionary can be implemented in the following way:

```
d={'id':1, 'name':"Aaki"}
for key, value in d.items():
    print(key, value)

= RESTART: C:\Users\DELL\Desktop\uni\fundamentals of computing\python programs\reportlaieg.py
id 1
name Aaki
```

Figure 16: Dictionary

In the above screenshot, I have declared a dictionary with 2 key-value pairs. Now using the for each loop the key and value items have been printed. The benefit of using dictionaries over list is that I don't have to remember the index in which "Aaki" is stored, instead I can use the key: name to access it.

2.4.2.4 Sets:

Set is also an unordered collection of data represented within curly brackets { }. The difference between set and dictionary is that set doesn't consist of key-value pairs. It is mutable, but cannot contain duplicate elements. They are unordered, hence they cannot be accessed through index, but for each loop can be used to gain access to each item through the set. The positions of elements within the set are random since indexes are not used.

Example: `s={1,2, "hello"}`

Use: Through different functions of sets such as `intersection()`, `union()`, `difference()`, `symmetricdifference()`, we can create a separate list according to our wish after comparison between two such sets using these functions. For example, `s= set1.intersection(set2)` stores the common elements present in both sets in a variable `s`.

Sets could be implemented in the following way:

```
s1={1, "hii", 4, 5}
s2={"hello", 3, 5}
a=s1.intersection(s2)
print(a)

=== RESTART: C:\Users\DELL\Desktop\uni\fundamentals of computing\python programs\reportlaieg.py ===
{5}
```

Figure 17: Set

In the above screenshots, two sets have been declared. The variable 'a' stores the value when we calculate the intersection between the 2 sets. Intersection refers to the common element present in both sets. When the value of 'a' is printed, the common element i.e. 5 in the form of set is printed.

3. Program:

3.1 Implementation of the Program:

The program is a simple implementation of a wholesale skincare shop: WeCare Wholesale. It maintains the available products, updates the changes and provides the suitable bill. The program was divided into 4 modules:

- i. main.py
- ii. read.py
- iii. write.py
- iv. operations.py

i. main.py:

This module is the main module of the program. It is mainly focused with importing the required functions with other modules. Along with that, the main function of this module is to correctly call the imported functions and execute the implementation process. Without this main module, the functions created in other modules would be of no use, since they would not have their implementation.

ii. read.py:

The read module contained function which read all details from the .txt file containing product information. Without this module, the product information stored in the .txt file would not be retrieved, and hence the further programming process would not carry on without the product data. Therefore, to get information from .txt file, this module has been used.

iii. write.py:

The write module contained all set of codes that were to be displayed to the user, or written to a unique file. The shop details, available stock, price, free items, etc. were all displayed through functions in the write.py module. This module created a user-friendly interface, allowing the user to interact effectively with the messages displayed in the screen. This module was also used to write details into unique .txt file. The unique

files, containing user details and bill information was also displayed via the write.py module

iv. operations.py:

The operations module contained the main logic. The process including taking input from user, validating input, calculating free items, price, shipping costs, restock process, etc. were performed within various functions in the operations.py module. Without this module, no mathematical calculations, and user input would be possible. Operations to update the stock after purchase and sale, and all minor calculations were performed in this module

WeCare.txt:

The text file WeCare.txt contained the product details. The read.py module retrieved product data from this file, and operations were performed on the basis of the data present here. The file was also updated after every purchase or sale, hence creating a real-life scenario system having stock update.

3.2 Complete process of purchase and sell:

3.2.1 Initial .txt file

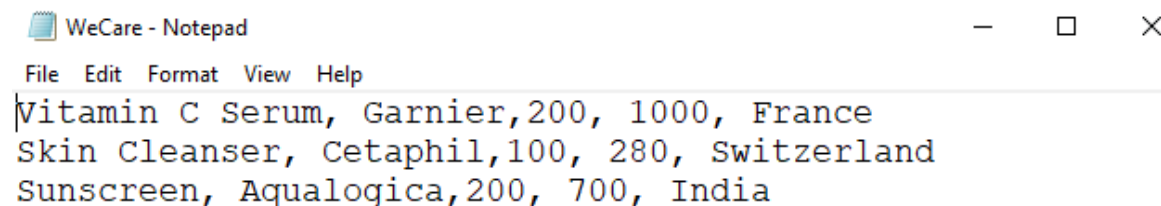


Figure 18: Initial .txt file

3.2.2 Sale process:



```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DELL\Desktop\uni\fundamentals of computing\python programs\24046683 Aaki Prajap
ati\24046683 Aaki Prajapati main.py

                WeCare Wholesale
                Lazimpat, Kathmandu
                Contact:01-4000021 Email:wecare@gmail.com

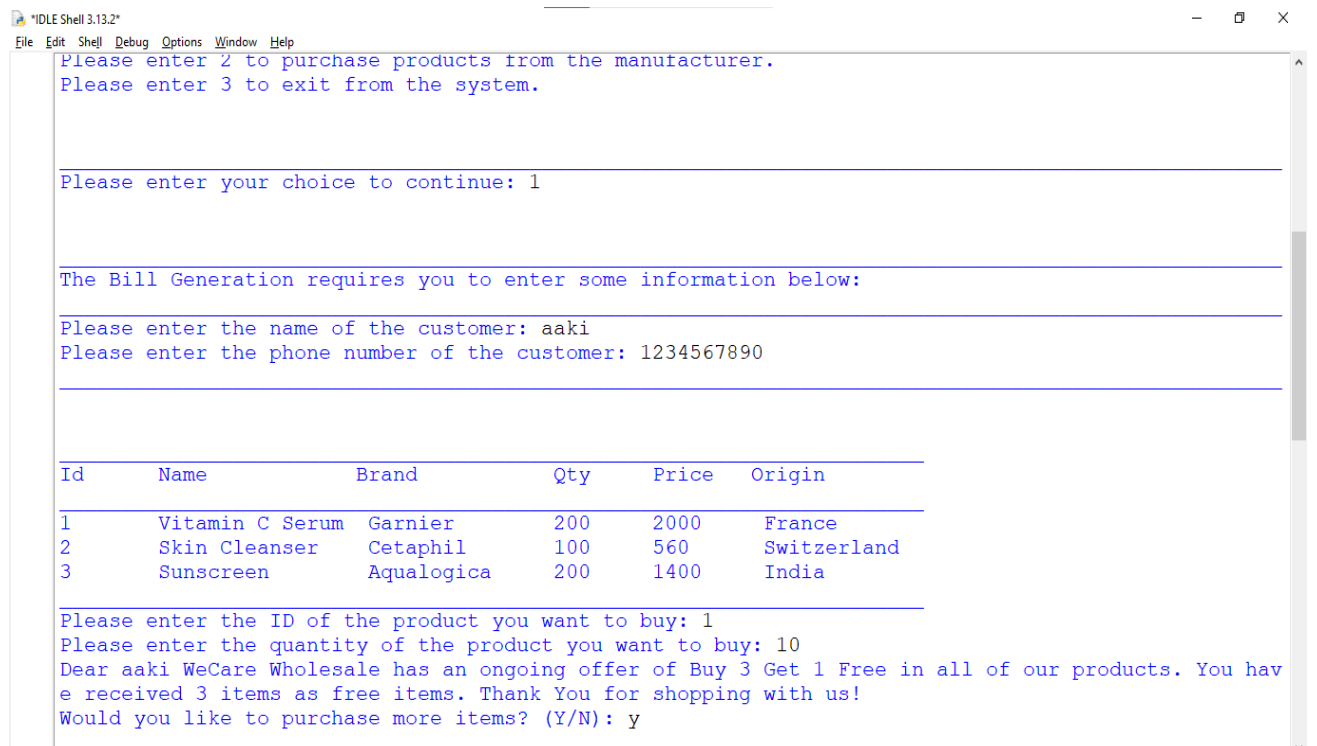
Dear Customer, Welcome to WeCare Wholesale! Hope you have a wonderful time with us.

Id      Name      Brand      Qty      Price      Origin
1       Vitamin C Serum  Garnier    200      1000      France
2       Skin Cleanser    Cetaphil   100      280       Switzerland
3       Sunscreen        Aqualogica 200      700       India

Dear Admin, please choose any one option from the choices below to carry out the operations.

Please enter 1 to sell the products to the customer.
Please enter 2 to purchase products from the manufacturer.
Please enter 3 to exit from the system.
```

Figure 19: Initial display



```
Please enter 2 to purchase products from the manufacturer.
Please enter 3 to exit from the system.

Please enter your choice to continue: 1

The Bill Generation requires you to enter some information below:

Please enter the name of the customer: aaki
Please enter the phone number of the customer: 1234567890

Id      Name      Brand      Qty      Price      Origin
1       Vitamin C Serum  Garnier    200      2000      France
2       Skin Cleanser    Cetaphil   100      560       Switzerland
3       Sunscreen        Aqualogica 200      1400      India

Please enter the ID of the product you want to buy: 1
Please enter the quantity of the product you want to buy: 10
Dear aaki WeCare Wholesale has an ongoing offer of Buy 3 Get 1 Free in all of our products. You hav
e received 3 items as free items. Thank You for shopping with us!
Would you like to purchase more items? (Y/N): y
```

Figure 20: Choosing option 1 and Selling item 1

```

IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help
2      Skin Cleanser      Cetaphil      100      560      Switzerland
3      Sunscreen         Aqualogica    200      1400     India

Please enter the ID of the product you want to buy: 1
Please enter the quantity of the product you want to buy: 10
Dear aaki WeCare Wholesale has an ongoing offer of Buy 3 Get 1 Free in all of our products. You have received 3 items as free items. Thank You for shopping with us!
Would you like to purchase more items? (Y/N): y

Id      Name      Brand      Qty      Price      Origin
-----
1      Vitamin C Serum      Garnier      187      2000      France
2      Skin Cleanser      Cetaphil      100      560      Switzerland
3      Sunscreen         Aqualogica    200      1400     India

Please enter the ID of the product you want to buy: 2
Please enter the quantity of the product you want to buy: 8
Dear aaki WeCare Wholesale has an ongoing offer of Buy 3 Get 1 Free in all of our products. You have received 2 items as free items. Thank You for shopping with us!
Would you like to purchase more items? (Y/N): n
Do you want the items to be shipped to your location? (Y/N): y

WeCare Wholesale Bill
Lazimpat, Kathmandu
Contact:01-4000021 Email:wecare@gmail.com

CUSTOMER DETAILS

```

Figure 21: Selling item 2 with updated stock

```

IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help
Do you want the items to be shipped to your location? (Y/N): y

WeCare Wholesale Bill
Lazimpat, Kathmandu
Contact:01-4000021 Email:wecare@gmail.com

CUSTOMER DETAILS

Customer Name: aaki      Customer Phone No: 1234567890
Purchase date and time: 2025-05-13 22:22:05.079244

PURCHASE DETAILS

Item      Quantity      Free Items      Unit Price      Total
-----
Vitamin C Serum      10      3      $2000      $20000
Skin Cleanser      8      2      $560      $4480

Shipping Cost: $500
Grand Total: $24980

Dear Admin, please choose any one option from the choices below to carry out the operations.

Please enter 1 to sell the products to the customer.
Please enter 2 to purchase products from the manufacturer.

```

Figure 22: Bill generation- sale

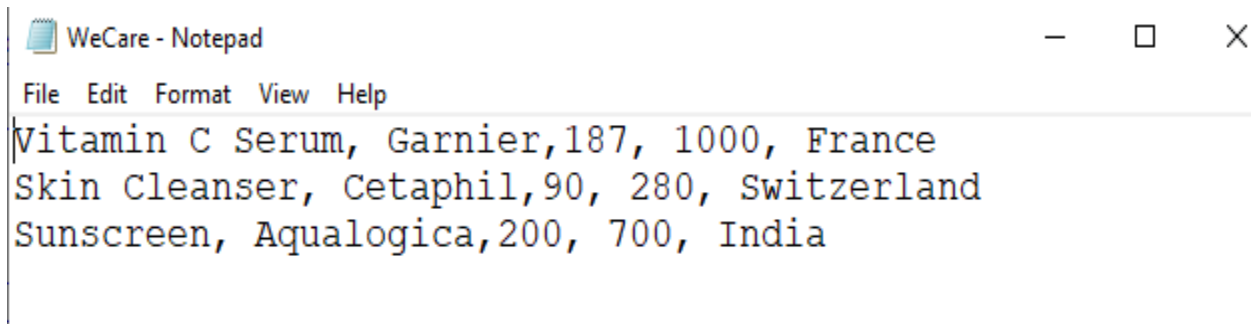


Figure 23: WeCare.txt file after sale

3.2.3 Purchase process:

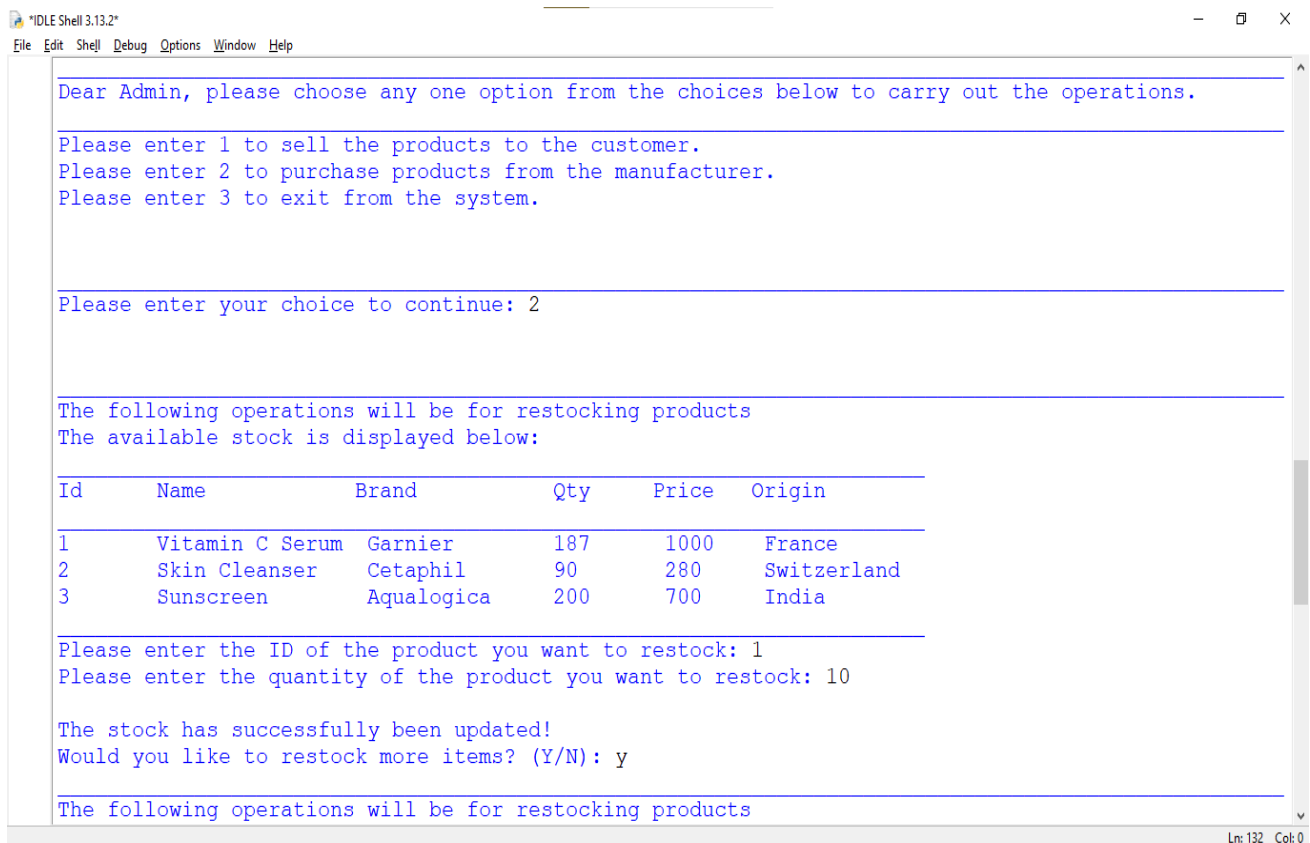
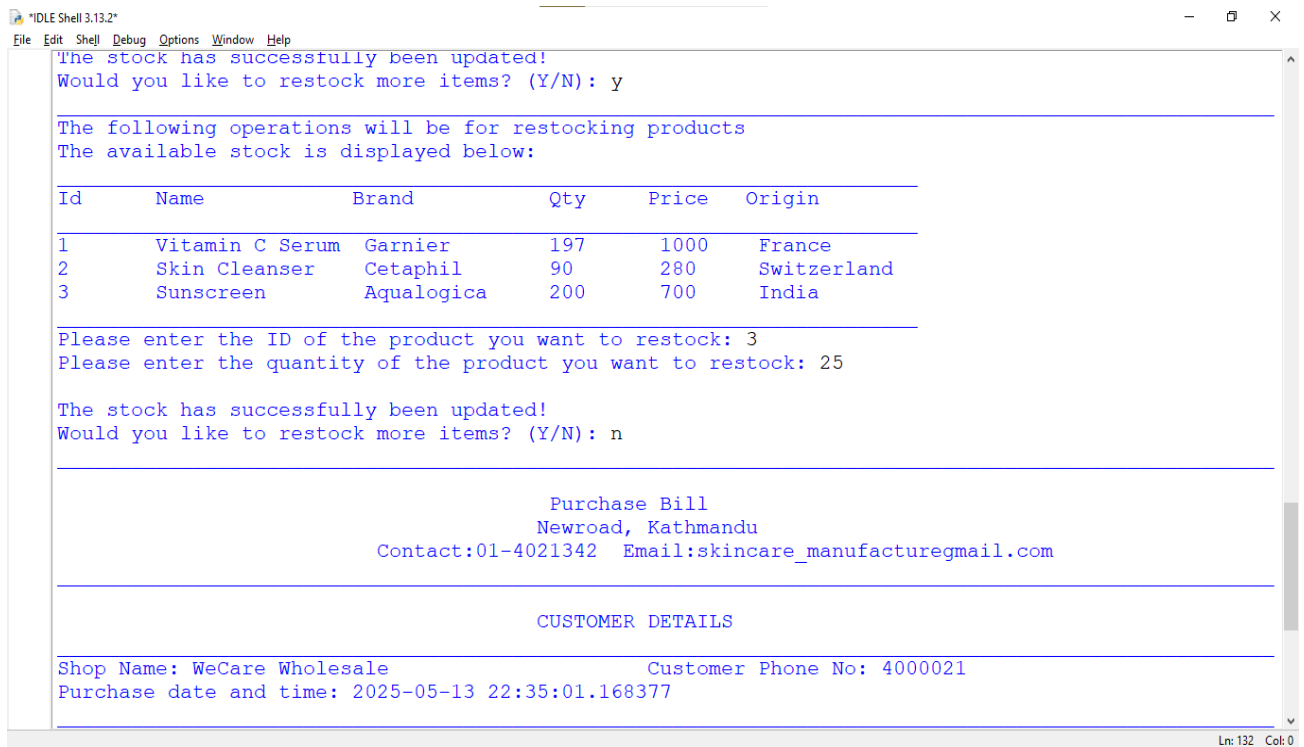


Figure 24: Choosing choice 2 and purchasing item 1



```

*IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help
The stock has successfully been updated!
Would you like to restock more items? (Y/N): y

The following operations will be for restocking products
The available stock is displayed below:

Id      Name          Brand      Qty      Price      Origin
-----
1       Vitamin C Serum    Garnier    197      1000       France
2       Skin Cleanser      Cetaphil   90       280        Switzerland
3       Sunscreen          Aqualogica 200      700        India

Please enter the ID of the product you want to restock: 3
Please enter the quantity of the product you want to restock: 25

The stock has successfully been updated!
Would you like to restock more items? (Y/N): n

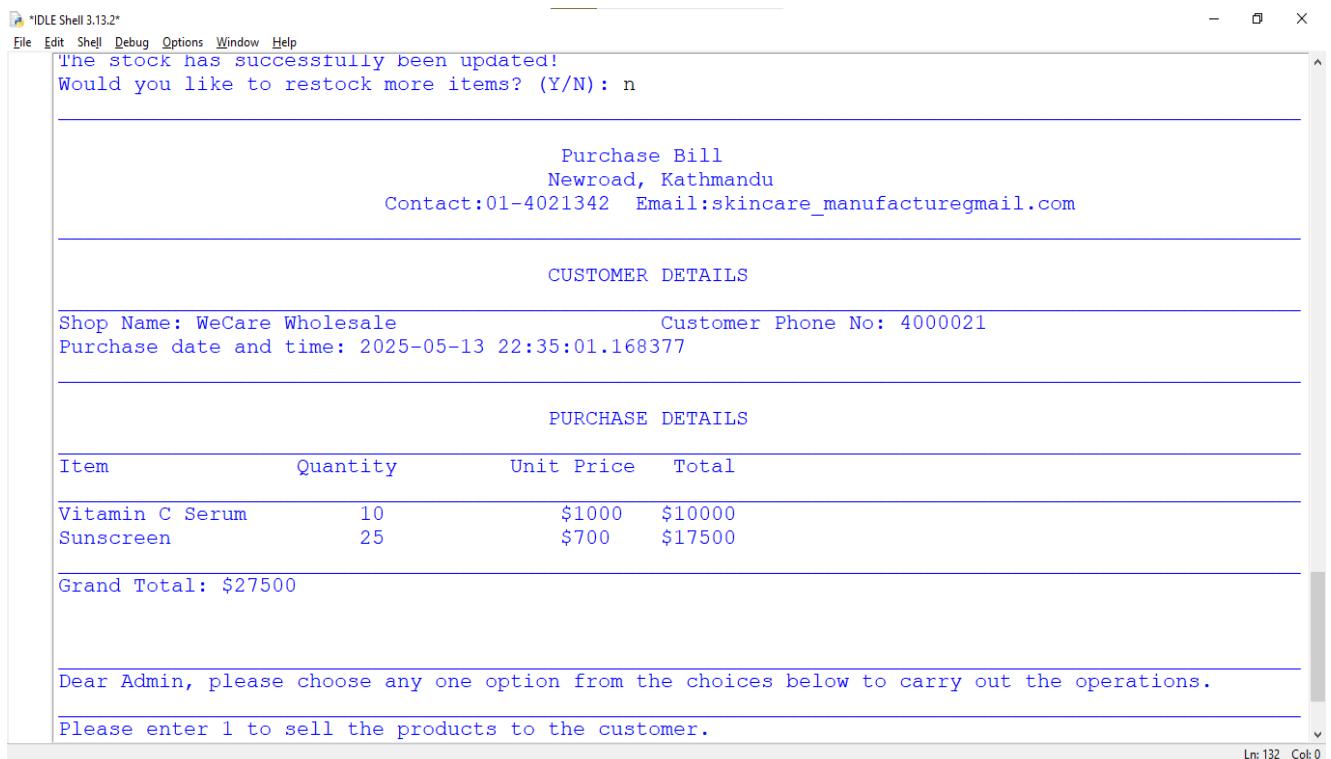
Purchase Bill
Newroad, Kathmandu
Contact:01-4021342 Email:skincare_manufacturegmail.com

CUSTOMER DETAILS

Shop Name: WeCare Wholesale      Customer Phone No: 4000021
Purchase date and time: 2025-05-13 22:35:01.168377

```

Figure 25: Purchasing item 3



```

*IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help
The stock has successfully been updated!
Would you like to restock more items? (Y/N): n

Purchase Bill
Newroad, Kathmandu
Contact:01-4021342 Email:skincare_manufacturegmail.com

CUSTOMER DETAILS

Shop Name: WeCare Wholesale      Customer Phone No: 4000021
Purchase date and time: 2025-05-13 22:35:01.168377

PURCHASE DETAILS

Item          Quantity      Unit Price      Total
-----
Vitamin C Serum    10          $1000          $10000
Sunscreen         25          $700           $17500

Grand Total: $27500

Dear Admin, please choose any one option from the choices below to carry out the operations.
Please enter 1 to sell the products to the customer.

```

Figure 26: Bill generation- purchase

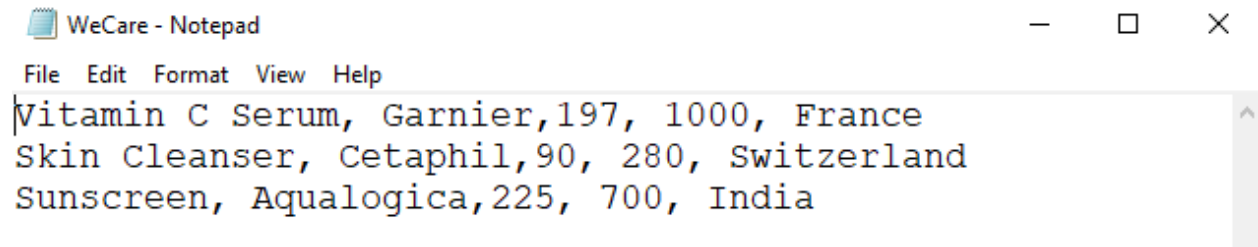


Figure 27: WeCare.txt after purchase

3.2 Creation of .txt file

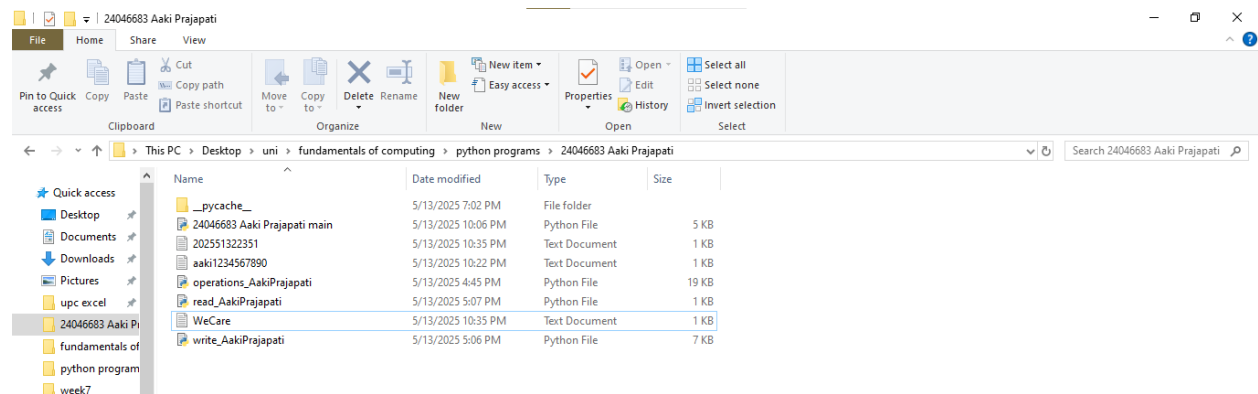


Figure 28: Creation of .txt files

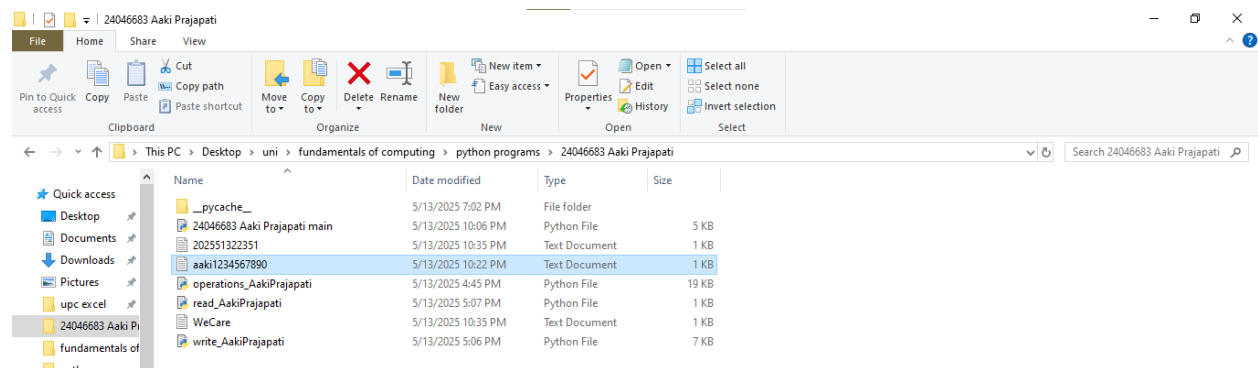


Figure 29: Creation of unique .txt file after sale

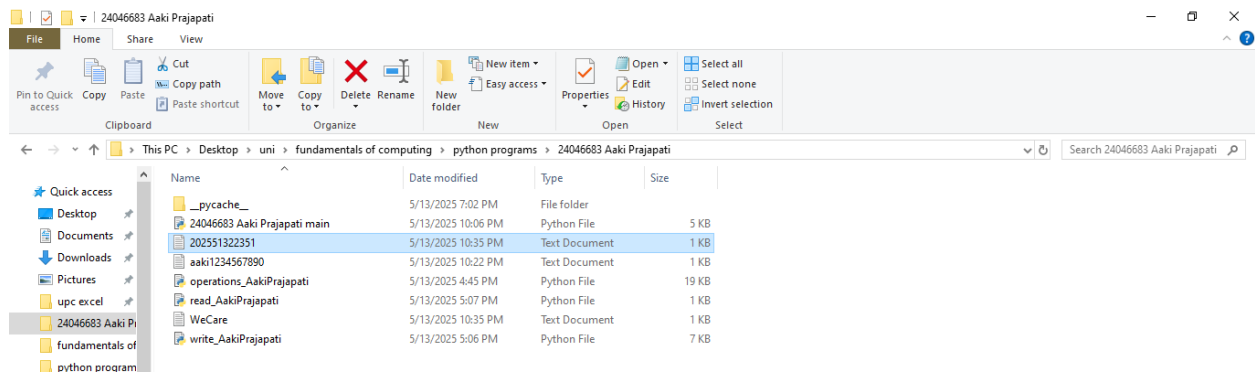


Figure 30: Creation of .txt file after purchase

3.3 Opening .txt files to show the bill:

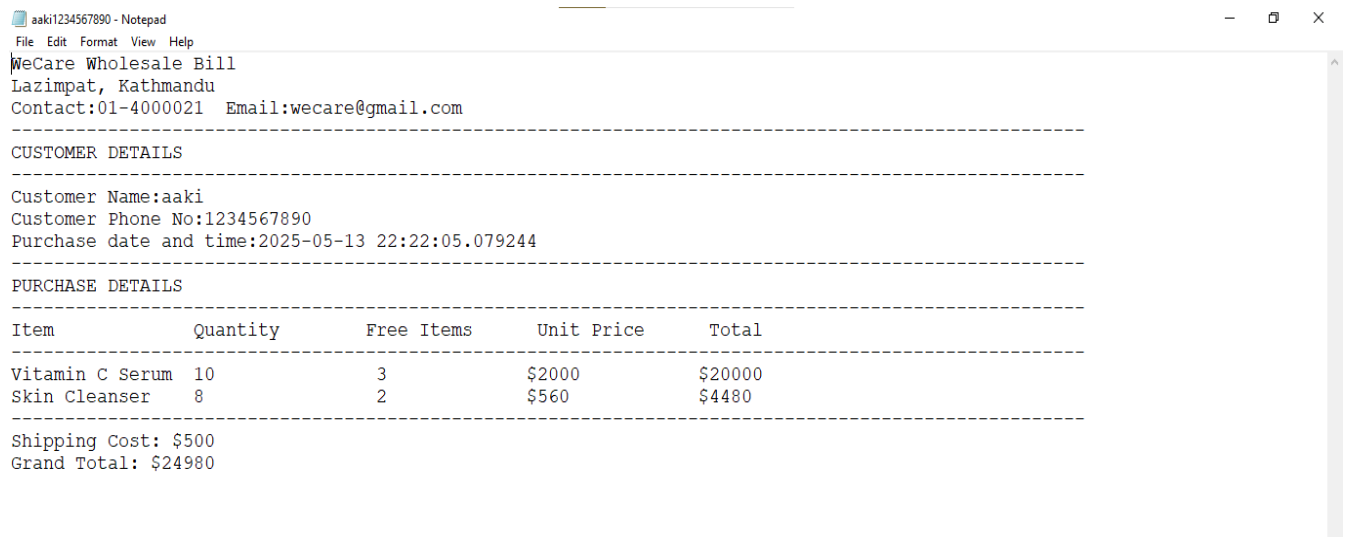


Figure 31: Bill generation- sale in unique file



Figure 32: Bill generation- purchase in unique file

3.4 Termination of Program:

```

IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help
Shop Name: WeCare Wholesale Customer Phone No: 4000021
Purchase date and time: 2025-05-13 22:35:01.168377

PURCHASE DETAILS
Item Quantity Unit Price Total
Vitamin C Serum 10 $1000 $10000
Sunscreen 25 $700 $17500
Grand Total: $27500

Dear Admin, please choose any one option from the choices below to carry out the operations.
Please enter 1 to sell the products to the customer.
Please enter 2 to purchase products from the manufacturer.
Please enter 3 to exit from the system.

Please enter your choice to continue: 3

Thank You for choosing us. Have a great day!
>>>
  
```

Figure 33: Choosing choice 3

4. Testing:

4.1 Test 1

Table 1: To check try and except block

Objective	To provide an invalid input and display the message.
Action	A string value was entered instead of integer when the user was asked to enter a choice.
Expected Result	An error message should be displayed.
Actual Result	An error message was displayed.
Conclusion	The test was successful.


```
Dear Admin, please choose any one option from the choices below to carry out the operations.
```

```
Please enter 1 to sell the products to the customer.
Please enter 2 to purchase products from the manufacturer.
Please enter 3 to exit from the system.
```

```
Please enter your choice to continue: hii
Please enter a number!
Please enter your choice to continue: |
```

Ln: 31 Col: 38

Figure 34: Implementation of try and except block

4.1 Test 2

4.1.1 Sale:

Table 2: To check negative and non-existent value as input (sale)

Objective	To provide a negative and a non-existent value as input one at a time.
Action	A negative value at first, followed by non-existent value was entered for ID of product to sell.
Expected Result	An invalid input message should be displayed both times.
Actual Result	An invalid input message was displayed both times.
Conclusion	The test was successful.

```
Id      Name      Brand      Qty      Price      Origin
1       Vitamin C Serum  Garnier    200      2000      France
2       Skin Cleanser    Cetaphil   100      560       Switzerland
3       Sunscreen        Aqualogica 200      1400      India

Please enter the ID of the product you want to buy: -8
Sorry! The ID you have entered is invalid. Please enter a valid ID!
Please enter the ID of the product you want to buy: 5
Sorry! The ID you have entered is invalid. Please enter a valid ID!
Please enter the ID of the product you want to buy: |
```

Ln: 53 Col: 52

Figure 35: Invalid input message displayed for negative and non-existent values (sale)

4.1.2 Purchase:

Table 3: To check negative and non-existent value as input (purchase)

Objective	To provide a negative and a non-existent value as input one at a time.
Action	A negative value at first, followed by non-existent value was entered for ID of product to sell.
Expected Result	An invalid input message should be displayed both times.
Actual Result	An invalid input message was displayed both times.
Conclusion	The test was successful.

```

Dear Admin, please choose any one option from the choices below to carry out the operations.

Please enter 1 to sell the products to the customer.
Please enter 2 to purchase products from the manufacturer.
Please enter 3 to exit from the system.

Please enter your choice to continue: 2

The following operations will be for restocking products
The available stock is displayed below:

Id      Name      Brand      Qty      Price      Origin
1       Vitamin C Serum  Garnier    200      1000      France
2       Skin Cleanser    Cetaphil   100      280       Switzerland
3       Sunscreen        Aqualogica 200      700       India

Please enter the ID of the product you want to restock: -5
Sorry! Invalid product ID! Please enter the correct product ID

Please enter the ID of the product you want to restock: 7
Sorry! Invalid product ID! Please enter the correct product ID

Please enter the ID of the product you want to restock: |

```

Figure 36: Error message displayed for negative and non-existent values (purchase)

4.3 Test 3

Table 4: To show file generation of multiple product purchase

Objective	To purchase multiple products and create one unique file to store purchase details.
Action	During purchase, multiple products were purchased and a unique file was created.
Expected Result	A unique file with purchase details should be created.
Actual Result	A unique file with purchase details was created.
Conclusion	The test was successful.

```

IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help

Please enter 1 to sell the products to the customer.
Please enter 2 to purchase products from the manufacturer.
Please enter 3 to exit from the system.

Please enter your choice to continue: 2

The following operations will be for restocking products
The available stock is displayed below:

Id      Name          Brand      Qty    Price  Origin
-----
1       Vitamin C Serum   Garnier    200    1000   France
2       Skin Cleanser     Cetaphil   100     280   Switzerland
3       Sunscreen         Aqualogica 200     700   India

Please enter the ID of the product you want to restock: 1
Please enter the quantity of the product you want to restock: 10

The stock has successfully been updated!
Would you like to restock more items? (Y/N): y

The following operations will be for restocking products
The available stock is displayed below:

```

Figure 37: Purchasing item 1

```

IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help

The following operations will be for restocking products
The available stock is displayed below:

Id      Name      Brand      Qty      Price      Origin
1      Vitamin C Serum  Garnier    210      1000      France
2      Skin Cleanser    Cetaphil   100      280       Switzerland
3      Sunscreen        Aqualogica 200      700       India

Please enter the ID of the product you want to restock: 2
Please enter the quantity of the product you want to restock: 5

The stock has successfully been updated!
Would you like to restock more items? (Y/N): y

The following operations will be for restocking products
The available stock is displayed below:

Id      Name      Brand      Qty      Price      Origin
1      Vitamin C Serum  Garnier    210      1000      France
2      Skin Cleanser    Cetaphil   105      280       Switzerland
3      Sunscreen        Aqualogica 200      700       India

Please enter the ID of the product you want to restock: 3
Please enter the quantity of the product you want to restock: 25

The stock has successfully been updated!
  
```

Figure 38: Purchasing item 2 and 3

```

IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help

Please enter the ID of the product you want to restock: 3
Please enter the quantity of the product you want to restock: 25

The stock has successfully been updated!
Would you like to restock more items? (Y/N): n

Purchase Bill
Newroad, Kathmandu
Contact:01-4021342 Email:skincare_manufacturegmail.com

CUSTOMER DETAILS

Shop Name: WeCare Wholesale Customer Phone No: 4000021
Purchase date and time: 2025-05-13 23:49:06.830121

PURCHASE DETAILS

Item      Quantity      Unit Price      Total
Vitamin C Serum    10      $1000      $10000
Skin Cleanser      5      $280      $1400
Sunscreen          25      $700      $17500

Grand Total: $28900
  
```

Figure 39: Bill Generation in shell



```

202551323497 - Notepad
File Edit Format View Help
Purchase Bill
Newroad, Kathmandu
Contact:01-4021342 Email:skincare_manufacturegmail.com
-----
CUSTOMER DETAILS
-----
Shop Name: WeCare Wholesale
Phone No: 4000021
Purchase date and time:2025-05-13 23:49:06.830121
-----
PURCHASE DETAILS
-----
Item          Quantity    Unit Price    Total
-----
Vitamin C Serum  10          $1000         $10000
Skin Cleanser    5           $280          $1400
Sunscreen        25          $700          $17500
-----
Grand Total: $28900

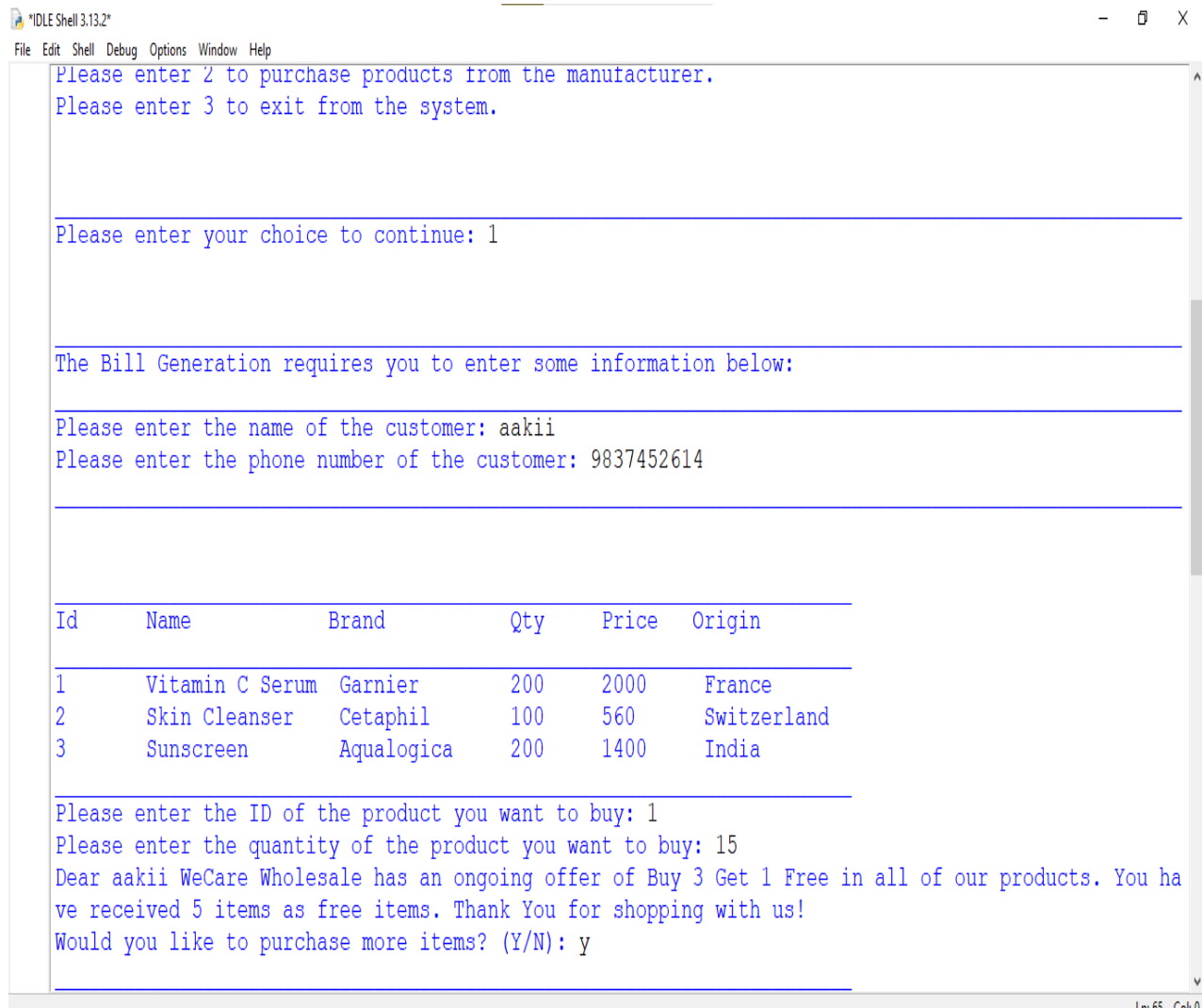
```

Figure 40: Purchased product details in .txt file

4.4 Test 4

Table 5: To show file generation of multiple product sale

Objective	To sell multiple products and create one unique file to store sale details.
Action	During sale, multiple products were sold and a unique file was created.
Expected Result	A unique file with sale details should be created.
Actual Result	A unique file with sale details was created.
Conclusion	The test was successful.



```
*IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help

Please enter 2 to purchase products from the manufacturer.
Please enter 3 to exit from the system.

Please enter your choice to continue: 1

The Bill Generation requires you to enter some information below:

Please enter the name of the customer: aakii
Please enter the phone number of the customer: 9837452614

Id      Name          Brand      Qty    Price  Origin
1      Vitamin C Serum  Garnier    200    2000   France
2      Skin Cleanser    Cetaphil   100     560   Switzerland
3      Sunscreen        Aqualogica 200    1400   India

Please enter the ID of the product you want to buy: 1
Please enter the quantity of the product you want to buy: 15
Dear aakii WeCare Wholesale has an ongoing offer of Buy 3 Get 1 Free in all of our products. You have received 5 items as free items. Thank You for shopping with us!
Would you like to purchase more items? (Y/N): y
```

Figure 41: Selling item 1

```

*IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help
Dear aakii WeCare Wholesale has an ongoing offer of Buy 3 Get 1 Free in all of our products. You ha
ve received 5 items as free items. Thank You for shopping with us!
Would you like to purchase more items? (Y/N): y

Id      Name      Brand      Qty      Price      Origin
-----
1       Vitamin C Serum  Garnier    180      2000      France
2       Skin Cleanser    Cetaphil   100      560       Switzerland
3       Sunscreen        Aqualogica 200      1400      India

Please enter the ID of the product you want to buy: 3
Please enter the quantity of the product you want to buy: 10
Dear aakii WeCare Wholesale has an ongoing offer of Buy 3 Get 1 Free in all of our products. You ha
ve received 3 items as free items. Thank You for shopping with us!
Would you like to purchase more items? (Y/N): n
Do you want the items to be shipped to your location? (Y/N): y

                                WeCare Wholesale Bill
                                Lazimpat, Kathmandu
                                Contact:01-4000021 Email:wecare@gmail.com

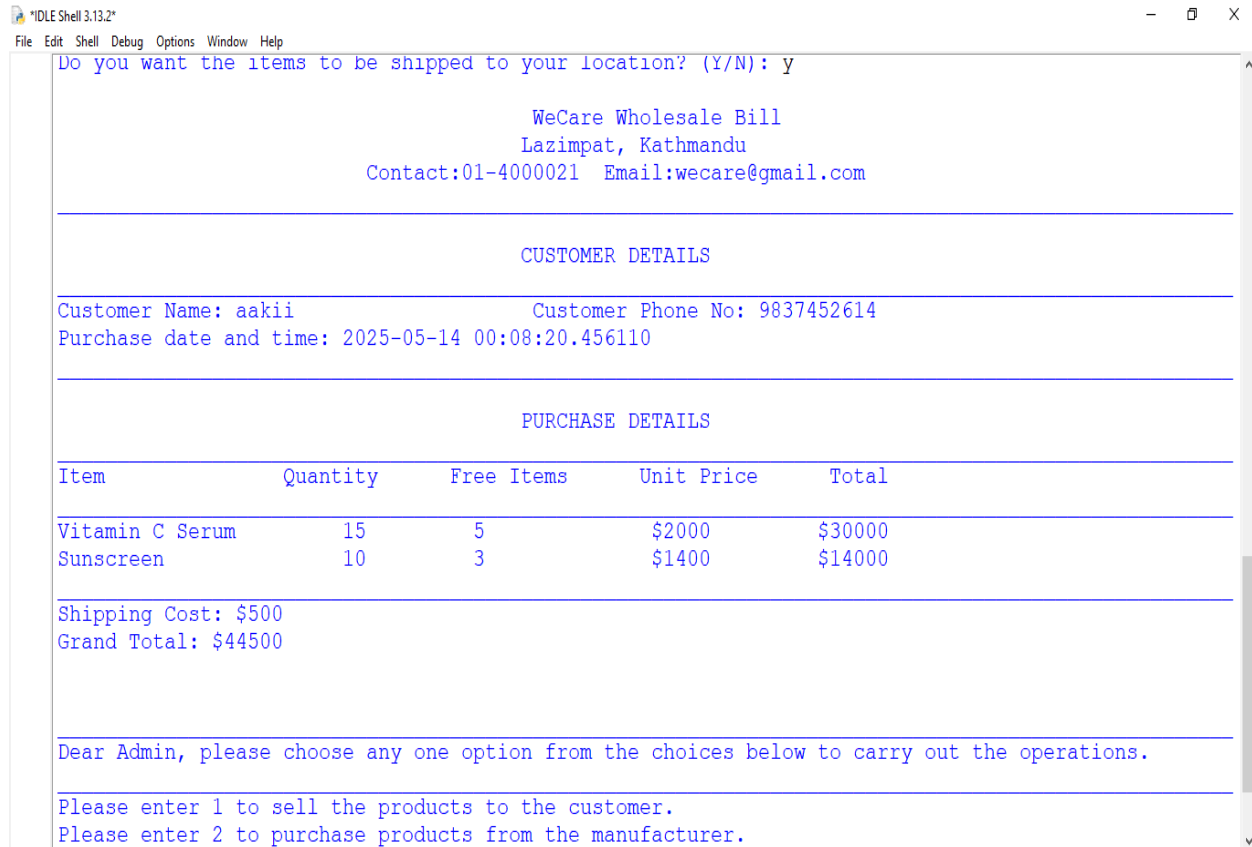
                                CUSTOMER DETAILS

Customer Name: aakii              Customer Phone No: 9837452614
Purchase date and time: 2025-05-14 00:08:20.456110

                                PURCHASE DETAILS

```

Figure 42: Selling item 2



```
*IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help
Do you want the items to be shipped to your location? (Y/N): y

                                WeCare Wholesale Bill
                                Lazimpat, Kathmandu
                                Contact:01-4000021 Email:wecare@gmail.com

                                CUSTOMER DETAILS

Customer Name: aakii                Customer Phone No: 9837452614
Purchase date and time: 2025-05-14 00:08:20.456110

                                PURCHASE DETAILS

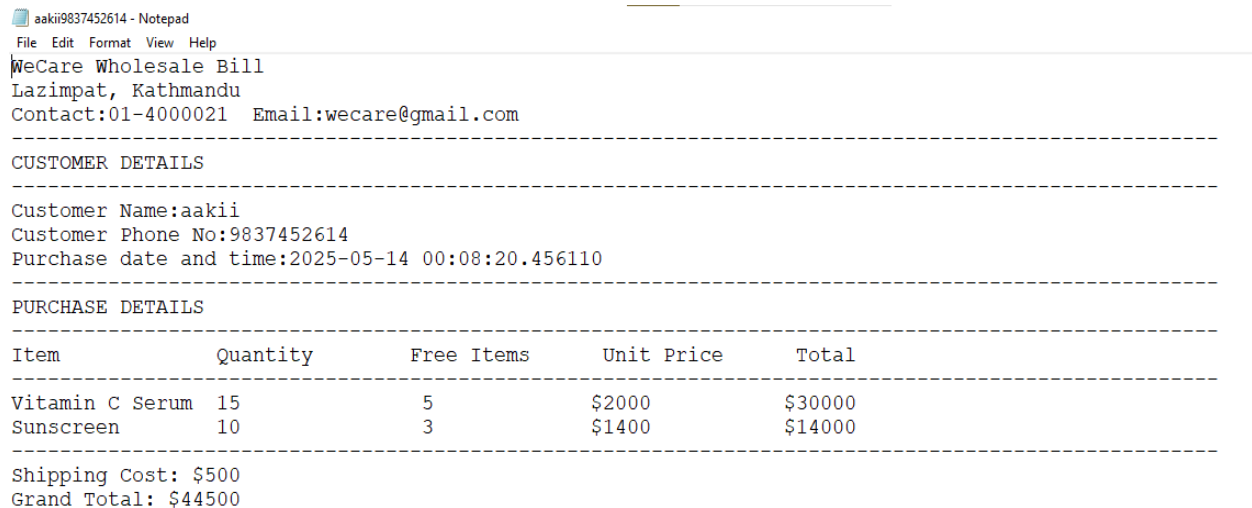
Item                Quantity    Free Items    Unit Price    Total
Vitamin C Serum     15                5          $2000      $30000
Sunscreen            10                3          $1400      $14000

Shipping Cost: $500
Grand Total: $44500

Dear Admin, please choose any one option from the choices below to carry out the operations.

Please enter 1 to sell the products to the customer.
Please enter 2 to purchase products from the manufacturer.
```

Figure 43: Bill generation in shell



```
aakii9837452614 - Notepad
File Edit Format View Help
WeCare Wholesale Bill
Lazimpat, Kathmandu
Contact:01-4000021 Email:wecare@gmail.com

-----
CUSTOMER DETAILS
-----
Customer Name:aakii
Customer Phone No:9837452614
Purchase date and time:2025-05-14 00:08:20.456110

-----
PURCHASE DETAILS
-----
Item                Quantity    Free Items    Unit Price    Total
Vitamin C Serum     15                5          $2000      $30000
Sunscreen            10                3          $1400      $14000

-----
Shipping Cost: $500
Grand Total: $44500
```

Figure 44: Sale product details in .txt file

4.5 Test 5

Table 6: To update the stock of products

Objective	To update the stock of products in shell as well as .txt file
Action	An item was sold and another item was purchased.
Expected Result	The stock should be updated at both items, in both shell and .txt file
Actual Result	The stock was updated at both items, in both shell and .txt file
Conclusion	The test was successful.

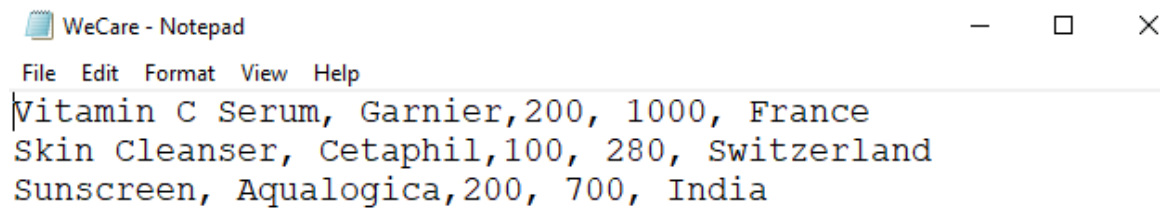
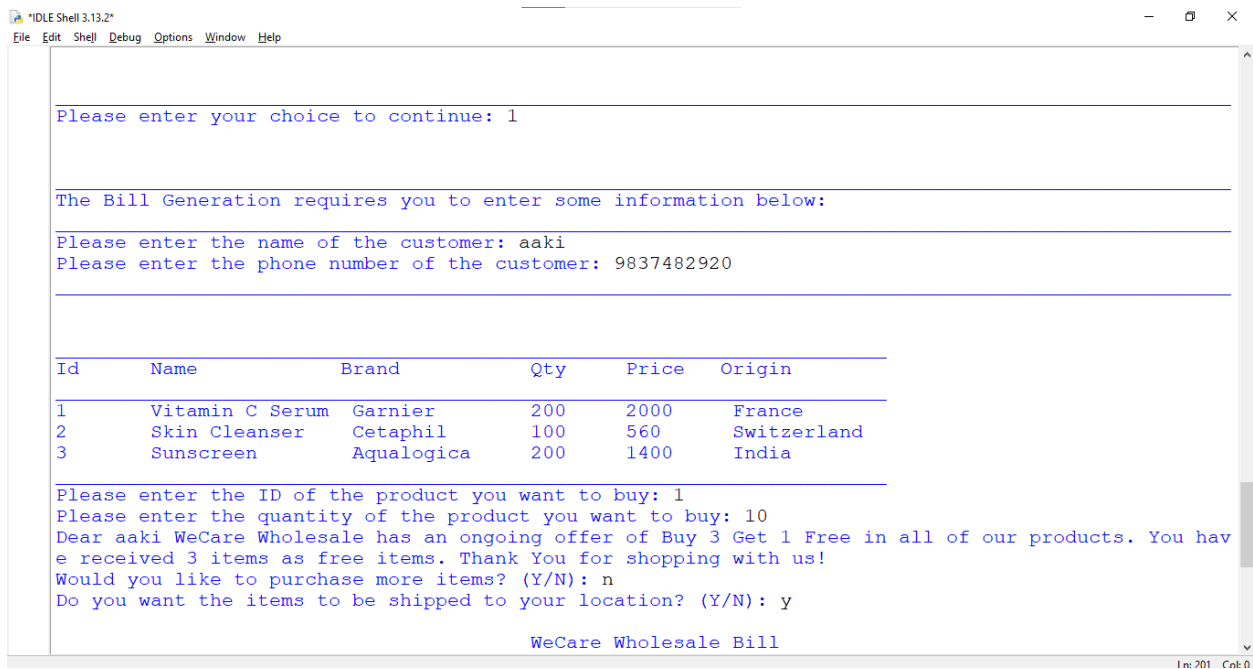


Figure 45: Initial .txt file



```

Please enter your choice to continue: 1

The Bill Generation requires you to enter some information below:

Please enter the name of the customer: aaki
Please enter the phone number of the customer: 9837482920

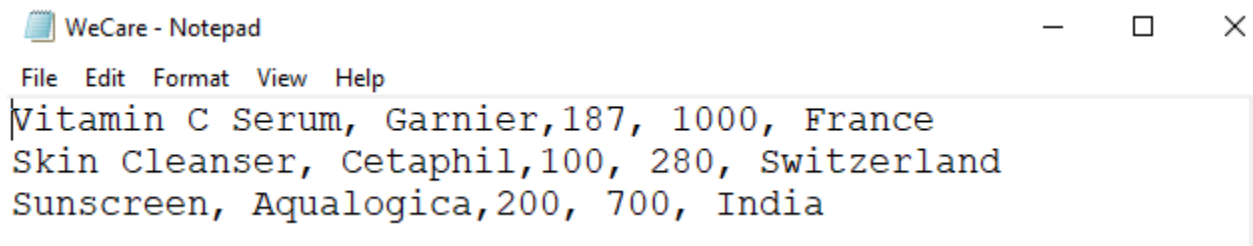
Id      Name      Brand      Qty      Price      Origin
1       Vitamin C Serum  Garnier    200      2000      France
2       Skin Cleanser    Cetaphil   100      560       Switzerland
3       Sunscreen        Aqualogica 200      1400      India

Please enter the ID of the product you want to buy: 1
Please enter the quantity of the product you want to buy: 10
Dear aaki WeCare Wholesale has an ongoing offer of Buy 3 Get 1 Free in all of our products. You have received 3 items as free items. Thank You for shopping with us!
Would you like to purchase more items? (Y/N): n
Do you want the items to be shipped to your location? (Y/N): y

WeCare Wholesale Bill

```

Figure 46: Selling item 1



```

Vitamin C Serum, Garnier,187, 1000, France
Skin Cleanser, Cetaphil,100, 280, Switzerland
Sunscreen, Aqualogica,200, 700, India

```

Figure 47: .txt file after sale

```

IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help

Dear Admin, please choose any one option from the choices below to carry out the operations.

Please enter 1 to sell the products to the customer.
Please enter 2 to purchase products from the manufacturer.
Please enter 3 to exit from the system.

Please enter your choice to continue: 1

The Bill Generation requires you to enter some information below:

Please enter the name of the customer: sjaaj
Please enter the phone number of the customer: 1234567890

Id      Name          Brand      Qty      Price      Origin
1       Vitamin C Serum  Garnier    187      2000       France
2       Skin Cleanser    Cetaphil   100      560        Switzerland
3       Sunscreen        Aqualogica 200      1400       India

Please enter the ID of the product you want to buy: |

```

Figure 48: Decreased quantity after sale

```

IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help

Dear Admin, please choose any one option from the choices below to carry out the operations.

Please enter 1 to sell the products to the customer.
Please enter 2 to purchase products from the manufacturer.
Please enter 3 to exit from the system.

Please enter your choice to continue: 2

The following operations will be for restocking products
The available stock is displayed below:

Id      Name          Brand      Qty      Price      Origin
1       Vitamin C Serum  Garnier    187      1000       France
2       Skin Cleanser    Cetaphil   100      280        Switzerland
3       Sunscreen        Aqualogica 200      700        India

Please enter the ID of the product you want to restock: 2
Please enter the quantity of the product you want to restock: 10

The stock has successfully been updated!
Would you like to restock more items? (Y/N): n

```

Figure 49: Purchasing item 2

```

IDLE Shell 3.13.2*
File Edit Shell Debug Options Window Help

Skin Cleanser          10          $280    $2800

Grand Total: $2800

Dear Admin, please choose any one option from the choices below to carry out the operations.

Please enter 1 to sell the products to the customer.
Please enter 2 to purchase products from the manufacturer.
Please enter 3 to exit from the system.

Please enter your choice to continue: 2

The following operations will be for restocking products
The available stock is displayed below:

Id      Name          Brand          Qty      Price  Origin
1       Vitamin C Serum  Garnier        187      1000   France
2       Skin Cleanser    Cetaphil       110      280    Switzerland
3       Sunscreen        Aqualogica     200      700    India

Please enter the ID of the product you want to restock:

```

Figure 50: Increased quantity after purchase

```

WeCare - Notepad
File Edit Format View Help
Vitamin C Serum, Garnier,187, 1000, France
Skin Cleanser, Cetaphil,110, 280, Switzerland
Sunscreen, Aqualogica,200, 700, India

```

Figure 51: .txt file after purchase

5. Conclusion:

In conclusion, through this coursework we learnt a lot more about programming with Python. It taught us to make use of different concepts within one program, such as 2D lists, modules, functions, loops, datatypes and much more. By implementing the “Buy 3 get 1 free” offer, we also got an idea of how business strategies are applied in real life to attract customers. Through the opportunity provided to us to develop a wholesale system, we got to implement the concept of file handling as well. This coursework provided us a platform to implement our knowledge to generate bills, maintain stock, keep track of records and much more. It also taught us about the importance of unique bill generation for each transaction, may it be purchase or sell. Small things such as maintaining a well-updated stock, producing invoices not only increased my ability to code, but also to think about it's real life importance. Although there were many difficulties that came along the way during the completion of this project, the determination overcame all of the difficulties. At last, with all of my effort, this coursework assigned to us by the Fundamentals of Computing department shows a user-friendly and well-responsive wholesale system.

6. Code Appendix:

6.1 read.py

```
#Reading and displaying the available stock
def read_from_file():
    """
    function: it is used to open the .txt file in read mode and store the items in a 2D
    list nl
    return: it returns the 2D list nl
    """
    file=open("WeCare.txt","r")    #opening the text file in read
    nl=[]
    lines=file.readlines()        #converts each line into a single list separated by \n
    for each in lines:            #for one line at a time from the list
        l=each.replace("\n","").split(",")    #replaces \n by a space and to identify
        separate elements of each line, it is split using , ""
        nl.append(l)              #storing all of the lists in one single list
    file.close()                  #to close the opened file
    return nl
```

6.2 write.py:

```
def initial_shop_details():
    """
    function: it is used to display the initial shop details
    """
    print("\n\t\t\t\t\tWeCare Wholesale")
    print("\t\t\t\t\tLazimpat, Kathmandu")
    print("\t\t\t\t\tContact:01-4000021 Email:wecare@gmail.com")
    print("_"*99)
    print("\n\tDear Customer, Welcome to WeCare Wholesale! Hope you have a
wonderful time with us.")
```

```
print("_"*99)

def user_display():
    """
    function: it is used to display available choices
    """
    print("_"*99)
    print("Dear Admin, please choose any one option from the choices below to carry out the operations.")
    print("_"*99)
    print("Please enter 1 to sell the products to the customer.")
    print("Please enter 2 to purchase products from the manufacturer.")
    print("Please enter 3 to exit from the system.")
    print("\n")
    print("_"*99)

def bill_info():
    """
    function: it is used to display a message to the user
    """
    print("_"*99)
    print("The Bill Generation requires you to enter some information below:")
    print("_"*99)

def display_free_items(name,free_products):
    """
    function: it is used to display the free items received by the user
    """
    print("Dear", name, "WeCare Wholesale has an ongoing offer of Buy 3 Get 1 Free in all of our products. You have received", free_products, "items as free items. Thank You for shopping with us!")
```

```

def update_after_purchase(nl):
    """
    parameter:the 2D list nl
    function: it is used to update the stock in file
    """
    file=open("WeCare.txt","w")
    for i in range(len(nl)):
        for j in range(len(nl[i])):
            file.write(nl[i][j])
            if j!=len(nl[i])-1:      #only if j is not the last element
                file.write(",")      #separate each element by ,
            file.write("\n")          #new line after the end of a product
    file.close()

def purchase_bill_onscreen(name, phone_no, date_and_time_tdy):
    """
    parameter: the customer name, phone number and current date and time
    function: it is used to display the shop details, and customer details
    """

    print("\n\t\t\t\t\tWeCare Wholesale Bill")
    print("\t\t\t\t\tLazimpat, Kathmandu")
    print("\t\t\t\t\tContact:01-4000021 Email:wecare@gmail.com")
    print("_"*99)
    print("\n\t\t\t\t\tCUSTOMER DETAILS")
    print("_"*99)
    print("Customer Name:",name,"\t\t\tCustomer Phone No:",phone_no)
    print("Purchase date and time:", str(date_and_time_tdy))
    print("_"*99)
    print("\n\t\t\t\t\tPURCHASE DETAILS")

```



```

print("_"*99)
print("Item \t\t Quantity\t Free Items\t Unit Price\t Total")
print("_"*99)

def restock_bill_onscreen(date_and_time_tdy):
    """
    parameter: the current date and time
    function: it is used to display _ for suitable representation
    """

    print("_"*99)

    #Displaying the final restock bill on the screen
    """
    function: it is used to display the shop details and other bill details
    """

    print("\n\t\t\t\t\tPurchase Bill")
    print("\t\t\t\t\t Newroad, Kathmandu")
    print("\t\t\t\t\t Contact:01-4021342 Email:skincare_manufacturegmail.com")
    print("_"*99)
    print("\n\t\t\t\t\t CUSTOMER DETAILS")
    print("_"*99)
    print("Shop Name: WeCare Wholesale","\t\t\tCustomer Phone No: 4000021")
    print("Purchase date and time:", str(date_and_time_tdy))
    print("_"*99)
    print("\n\t\t\t\t\t PURCHASE DETAILS")
    print("_"*99)
    print("Item \t\t Quantity\t Unit Price\t Total")
    print("_"*99)

def unique_bill(name, phone_no, date_and_time_tdy, shipping_cost, grand_total,
selling_items):

```

```
"""
```

parameter: name, phone number, current date and time, shipping cost, total with shipping cost

and selling_items list

function: it is used to write bill details along with name, quantity, free items, unit price,

total of each item, total of all items along with shipping cost

```
"""
```

```
file=open(name+str(phone_no)+".txt","w")
```

```
file.write("WeCare Wholesale Bill")
```

```
file.write("\nLazimpat, Kathmandu")
```

```
file.write("\nContact:01-4000021 Email:wecare@gmail.com\n")
```

```
file.write("-"*100)
```

```
file.write("\nCUSTOMER DETAILS\n")
```

```
file.write("-"*100)
```

```
file.write("\nCustomer Name:"+name)
```

```
file.write("\nCustomer Phone No:"+phone_no)
```

```
file.write("\nPurchase date and time:"+ str(date_and_time_tdy)+ "\n")
```

```
file.write("-"*100)
```

```
file.write("\nPURCHASE DETAILS\n")
```

```
file.write("-"*100)
```

```
file.write("\nItem\t\t Quantity\t Free Items\t Unit Price\t Total\n")
```

```
file.write("-"*100)
```

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

#accessing through the selling_items list where all the purchase information is stored

```
for a in selling_items:
```

```
    file.write(a[0]+\t " +str(a[1])+ "\t\t " +str(a[2])+ "\t\t" + "$"+str(a[3])+ "\t\t" + "$"+
str(a[4])+"\n")
```

```
    file.write("-"*100)
```

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

```

if shipping_cost>0:
    file.write("Shipping Cost: $" +str(shipping_cost))
file.write("\nGrand Total: $" +str(grand_total))
file.close()

```

```

def file_after_restock(nl):
    """
    parameter: 2D list containing product data
    function: it is used to open the .txt file in write mode, and update it
    """
    file=open("WeCare.txt","w")
    for i in range(len(nl)):
        file.write(nl[i][0]+", "+nl[i][1]+", "+nl[i][2]+", "+nl[i][3]+", "+nl[i][4])
        file.write("\n")
    file.close()

```

#Bill after restock

```

def bill_after_restock(unique_num, total_of_all, purchase, date_and_time_tdy):
    """
    parameter: unique date and time, total price of all products, purchase list
               and date and time today
    function: to display the bill details
    """
    file=open(unique_num+".txt","w")
    file.write("Purchase Bill")
    file.write("\nNewroad, Kathmandu")
    file.write("\nContact:01-4021342 Email:skincare_manufacturegmail.com\n")
    file.write("-"*100)
    file.write("\nCUSTOMER DETAILS\n")
    file.write("-"*100)
    file.write("\nShop Name: WeCare Wholesale")

```

```

file.write("\nPhone No: 4000021")
file.write("\nPurchase date and time:"+ str(date_and_time_tdy)+"\n")
file.write("-"*100)
file.write("\nPURCHASE DETAILS\n")
file.write("-"*100)
file.write("\nItem\t\t Quantity\t Unit Price\t Total\n")
file.write("-"*100)
file.write("\n")
#accessing through the selling_items list where all the purchase information is
stored
for a in purchase:
    file.write(a[0]+\t " +str(a[1])+ "\t\t" + "$"+str(a[2])+ "\t\t" + "$"+ str(a[3])+"\n")
file.write("-"*100)
file.write("\nGrand Total: $"+str(total_of_all))
file.close()

def option_3():
    """
    function: it is used to display thankyou message
    """
    print("Thank You for choosing us. Have a great day!")

#Invalid choice
def invalid_choice(choices):
    """
    function: it is used to display sorry message
    """
    print("Sorry! The choice",choices,"does not exist. Please enter the correct
choice.\n")

```

6.3 operations.py

```

import datetime

def display_stock(nl):
    """
    parameter: the 2D list containing data from .txt file
    function: it is used to display the initial stock
    """

    product_id=1                #introducing product id as 1.
    print("_"*70)                #prints underscore 70 times
    print("Id\tName\tBrand\tQty\tPrice\tOrigin")
    print("_"*70)

    for i in range(len(nl)):      #for lists in nl
        print(product_id,end="\t")    #printing the product id of each product
        for j in range(len(nl[i])):  #for elements of list in nl
            print(nl[i][j],end="\t")  #print each element present within lists which are
#present in nl with space between each element using end=\t since by default each
#component is printed a new line
        print()                  #change lines after each list ends
        product_id+=1
    print("_"*70)

#taking choices as input from user
def input_choices():
    """
    function: it is used ask the user's choice
    return: to return the choice entered by the user
    """

    choice_loop=False
    while choice_loop==False:
        try:
            choices=int(input("Please enter your choice to continue: "))

```

```

        print("\n")
        choice_loop=True
    except:
        print("Please enter a number!")
        choice_loop=False
    return choices

```

#CHOICE: 1

#asking name and phone number

```
def enter_user_details():
```

```
    """
```

function: it is used to ask name and phone number, and restrict the phone number upto 10 digits

return: to return the name and phone no entered by user

```
    """
```

```
    name=input("Please enter the name of the customer: ")
```

```
    phone_no=input("Please enter the phone number of the customer: ")
```

```
    #Validating phone number
```

```
    while len(phone_no)<10 or len(phone_no)>10:
```

```
        print("Phone Number must be 10 digits!")
```

```
        phone_no=input("Please enter the phone number of the customer: ")
```

```
    print("_"*99)
```

```
    print("\n")
```

```
    return name, phone_no
```

```
def initialize_variable():
```

```
    """
```

function: it is used to initialize selling_items list, total, grand total, shipping cost and selling_loop

return: to return the variables that were initialized

```
    """
```

```

selling_items=[]      #A list to store the items purchased by the customer
total=0               #Total bill without shipping
grand_total=0         #The total cost to be paid by the customer
shipping_cost=0       #Shipping cost if the customer wants the products to be
shipped
selling_loop=True     #Selling loop to proceed the buying process
return selling_items, total, grand_total, shipping_cost, selling_loop

```

```
def buying_product_id(nl):
```

```
    """
```

```
    parameter: the 2D list containing product data
```

```
    function: it is used to ask ID of product that the user wants to buy, and validate it
```

```
    return: to return the validated ID of the product
```

```
    """
```

```
    buying_loop=False
```

```
    while buying_loop==False:
```

```
        try:
```

```
            #Taking Product ID from the customer
```

```
            buying_id=int(input("Please enter the ID of the product you want to buy: "))
```

```
            #Validating the entered ID
```

```
            buying_id=buying_id_validation(buying_id, nl)
```

```
            buying_loop=True
```

```
        except:
```

```
            print("Please enter a number!")
```

```
            buying_loop=False
```

```
    return buying_id
```

```
def buying_product_qty(nl):
```

```
    """
```

```
    parameter: 2D list containing product data
```

```
    function: it is used to ask the quantity of products that the user wants to buy
```

```

    return: quantity of products that the user wants to buy
    """

    purchase_quantity_loop=False
    while purchase_quantity_loop==False:
        try:
            product_quantity=int(input("Please enter the quantity of the product you want
to buy: "))
            purchase_quantity_loop=True
        except:
            print("Please enter a number!")
            purchase_quantity_loop=False
    return product_quantity

def purchase_product(product_quantity,nl,buying_id):
    """
    parameter: ID and product quantity entered by the user, 2D list containing product
data
    function: it is used to calculate number of free items, quantity to deduct from the
stock, to
        check if the product quantity entered by the user is valid, if not it asks the
user
        to re-enter the product quantity
    return: number of free items to be received by the customer, total quantity to be
deducted from
        stock and total quantity of products to be bought
    """

    free_products= product_quantity//3           #Gives the quotient without decimal
points\
    quantity_to_deduct= product_quantity+free_products  #Total number of products
that should be deducted, including the free items

```



```
#Getting the quantity of the product asked by the user
qty_of_selected_product = nl[buying_id - 1][2]
#The buying ID is subtracted by 1 because the product that the user is trying to get
through id is actually stored at an index -1 than the id in the 2D list
```

```
#Checking available stock- validating quantity
while product_quantity<=0 or quantity_to_deduct>int(qty_of_selected_product):
#If the quantity entered by user is <=0 or if the total number of items including the
#free items after the user has entered the product qty is more than the available
stock
```

```
    print("Sorry! The quantity as entered is not available in our shop. We request you
to kindly check the product table again, and enter the quantity")
```

```
    #Asking the user to enter the quantity and calculate the free products once again
    product_quantity=int(input("Please enter the quantity of the product you want to
buy: "))
```

```
    free_products= product_quantity//3          #Gives the quotient without decimal
points
```

```
    quantity_to_deduct= product_quantity+free_products #Total number of products
that should be deducted, including the free items
```

```
    print("\n")
    return free_products, quantity_to_deduct, product_quantity
```

```
def buy_more():
```

```
    """
```

```
function: it is used to ask the user if they want to purchase more items
```

```
return: the boolean value of loop based off of user's decision
```

```
    """
```

```
    buy_more=input("Would you like to purchase more items? (Y/N): ").lower()
```

```
    if buy_more=="y" or buy_more=="yes":
```

```
        selling_loop=True
```

```
    else:
```

```

    selling_loop=False
    return selling_loop

```

```

def stock_display_choice1(nl):
    """
    parameter: the 2D list containing data from .txt file
    function: it is used to display the stock when choice 1 is entered (sp=cp*2)
    """

    #Displaying the stock available
    print("_"*70)                #prints underscore 70 times
    print("Id\tName\t\tBrand\t\tQty\tPrice\tOrigin")
    print("_"*70)
    product_id=1                 #resetting product id to 1
    for i in range(len(nl)):      #for lists in nl
        print(product_id,end="\t")    #printing the product id of each product
        for j in range(len(nl[i])):  #for elements of list in nl
            if j==3:
                print(int(nl[i][j])*2,end="\t") #the selling price is twice the cost price
            else:
                print(nl[i][j],end="\t")    #print each element present within lists which are
present in nl with space between each element using end=\t since by default each
component is printed a new line
        print()                  #change lines after each list ends
        product_id+=1
    print("_"*70)

    #Validate ID entered by user
    def buying_id_validation(buying_id, nl):
        """
        parameter: ID of the product entered by the user, 2D list containing product data

```

```

function: it is used to check if the entered product ID is valid, if not it asks
the user to re-enter the product ID
return: it returns the validated ID of product
"""

#Checking if the product ID is valid
while buying_id<=0 or buying_id>len(nl): #Checking if the admin has entered an id
less than 0 or greater than the available product ids
    print("Sorry! The ID you have entered is invalid. Please enter a valid ID!")

    #After the invalid ID message has been displayed, asking the admin to enter the
ID once again
    buying_id=int(input("Please enter the ID of the product you want to buy: "))
    return buying_id

#Updating stock after purchase
def update_stock_after_purchase(buying_id, product_quantity, free_products,
quantity_to_deduct,selling_items, total, nl):
    """
    parameter: ID, product quantity, total number of free products, total quantity to be
deducted

    from stock, selling_items list, total price of items, 2D list nl
    function: it is used to update the previous stock by deducting the quantity bought
and free items,

    to access the name, unit price of the products,
    to calculate the total price of each item
    to add values to the list and calculate total price of all products
    return: selling_items loop and total price of all products
    """
    nl[buying_id - 1][2]=str(int(nl[buying_id - 1][2])-quantity_to_deduct)
    #reducing the previous stock with the quantity to deduct

```

```

#Bill Calculation
product_name=nl[buying_id - 1][0]      #The name of the required product is
stored in product_name
unit_price=int(nl[buying_id - 1][3])*2  #The price per piece of the required product
is stored in unit_price
#selling price is twice the cost price
total_price=unit_price*product_quantity  #total price of one product

#Storage of the purchased items
selling_items.append([product_name, product_quantity,free_products, unit_price,
total_price]) #Storing all information in a list
total=total+total_price                #total price of all products chosen by customer
return selling_items, total

def shipping(shipping_cost):
    """
    parameter: whether the user wants to ship or not, shipping cost
    function: it is used to add the shipping cost if the user wishes to have their products
    shipped
    return: to return the shipping cost
    """
    ship=input("Do you want the items to be shipped to your location? (Y/N): ").lower()
    if ship=="y" or ship=="yes":
        shipping_cost=500
    return shipping_cost

def final_total(grand_total, total, shipping_cost):
    """
    parameter: the grand total cost with shipping, total cost without shipping, shipping
    cost
    function: it is used to calculate the grand total price with shipping

```

```

    return: to return the grand total
    """

    grand_total=total+shipping_cost
    return grand_total

def date_time_tdy():
    """
    function:it is used to determine the current date and time
    return: to return the current date and time
    """

    date_and_time_tdy=datetime.datetime.now()
    return date_and_time_tdy

def bill_items(selling_items, shipping_cost, grand_total):
    """
    parameter: the selling_items list, shipping cost, and grand total cost with shipping
    function: to display the item details
    """

    for a in selling_items:
        print(a[0]+"\\t\\t" +str(a[1])+ "\\t    " +str(a[2])+ "\\t\\t    " +"$"+str(a[3])+ "\\t\\t" +"$"+
str(a[4]))
    print("_"*99)
    if shipping_cost>0:
        print("Shipping Cost: $"+str(shipping_cost))
    print("Grand Total: $"+str(grand_total))
    print("\\n")

#CHOICE: 2
#Displaying available stock

def variables():

```

```
"""  
  
function: it is used to initialize purchase loop, total_of_all, and purchase_loop  
return: to return the variables that have been initialized  
"""  
  
purchase=[]          #initializing before loop to add multiple products in a single bill  
total_of_all=0  
purchase_loop=True  
return purchase, total_of_all, purchase_loop  
  
def restock_id(nl):  
    """  
  
    parameter: 2D list containing product details  
    function: it is used to ask the user to enter the ID of product to be restocked and  
validate it  
    return: to return the validated ID of the product  
    """  
  
    restock_loop=False  
    while restock_loop==False:  
        try:  
            restock_product_id=int(input("Please enter the ID of the product you want to  
restock: "))  
            #Validating the product ID  
            restock_product_id=restock_id_validation(restock_product_id,nl)  
            restock_loop=True  
        except:  
            print("Please enter a number!")  
            restock_loop=False  
    return restock_product_id  
  
def restock_qty(nl):  
    """
```

```

parameter: 2D list containing product details
function: it is used to ask the user to enter the quantity of product to be restocked
and
        validate it
return: to return the quantity of the product to be restocked
"""

restock_quantity_loop=False
while restock_quantity_loop==False:
    try:
        restock_product_qty=int(input("Please enter the quantity of the product you
want to restock: "))
        #Validating the quantity of the product to be restocked
        restock_product_qty=restock_quantity_validation(restock_product_qty)
        restock_quantity_loop=True
    except:
        print("Please enter a number!")
        restock_quantity_loop=False
return restock_product_qty

def restock_again():
    """
    function: it is used to ask if the user wants to restock more items
    return: to return the boolean value of purchase_loop based off of the user's answer
    """

    print("\nThe stock has successfully been updated!")
    #Asking the user if they want to restock any more products
    restock_more=input("Would you like to restock more items? (Y/N): ").lower()
    if restock_more=="y" or restock_more=="yes":
        purchase_loop=True
    else:
        purchase_loop=False

```

```
return purchase_loop
```

```
def restock_bill_items(total_of_all, purchase):
```

```
    """
```

```
    parameter: the total price of all items, and purchase list
```

```
    function: it is used to display item details
```

```
    """
```

```
    #accessing through the purchase list where all the purchase information is stored
    for a in purchase:
```

```
        print(a[0]+"\\t\\t" +str(a[1])+ "\\t\\t" +"$" +str(a[2])+ "\\t" +"$" + str(a[3]))
```

```
    print("_"*99)
```

```
    print("Grand Total: $" +str(total_of_all))
```

```
    print("\\n")
```

```
def unique_date_time():
```

```
    """
```

```
    function: it is used to concatenate the year, month, day, hour, minute and second
    to create a unique number
```

```
    return: to the unique_num which is the current date and time
```

```
    """
```

```
    year=str(datetime.datetime.now().year)
```

```
    month=str(datetime.datetime.now().month)
```

```
    day=str(datetime.datetime.now().day)
```

```
    hour=str(datetime.datetime.now().hour)
```

```
    minute=str(datetime.datetime.now().minute)
```

```
    second=str(datetime.datetime.now().second)
```

```
    unique_num=year+month+day+hour+minute+second
```

```
    return unique_num
```

```
def stock_display_choice2(nl):
```

```
    """
```


parameter: the 2D list containing data from .txt file

function: it is used to display the initial stock

"""

product_id=1

print("_"*99)

print("The following operations will be for restocking products")

print("The available stock is displayed below:")

print("_"*70) #prints underscore 70 times

print("Id\tName\tBrand\tQty\tPrice\tOrigin")

print("_"*70)

for i in range(len(nl)): #for lists in nl

print(product_id,end="\t") #printing the product id of each product

for j in range(len(nl[i])): #for elements of list in nl

print(nl[i][j],end="\t") #print each element present within lists which are

present in nl with space between each element using end=\t since by default each component is printed a new line

print() #change lines after each list ends

product_id+=1

print("_"*70)

#Validating ID of the product to be restocked

def restock_id_validation(restock_product_id,nl):

"""

parameter: ID of the product entered by the user, 2D list containing product data

function: it is used to check if the entered product ID is valid, if not it asks

the user to re-enter the product ID to be restocked

return: it returns the validated ID of product

"""

while restock_product_id<=0 or restock_product_id>len(nl):

print("Sorry! Invalid product ID! Please enter the correct product ID\n")

#Asking the valid product ID from the user of the product they want to restock

```

        restock_product_id=int(input("Please enter the ID of the product you want to
restock: "))

```

```

    return restock_product_id

```

```

#Validating quantity of the product to be restocked

```

```

def restock_quantity_validation(restock_product_qty):

```

```

    """

```

```

    parameter: quantity of the product entered by the user

```

```

    function: it is used to check if the entered product quantity is valid, if not it asks
the user to re-enter the product quantity

```

```

    return: it returns the validated quantity of product

```

```

    """

```

```

    while restock_product_qty<=0:

```

```

        print("Sorry! You must restock atleast 1 item")

```

```

        restock_product_qty=int(input("Please enter the quantity of the product you want
to restock: "))

```

```

    return restock_product_qty

```

```

#Updating stock after restocking

```

```

def stock_after_restock(restock_product_id, restock_product_qty,nl):

```

```

    """

```

```

    parameter: ID of the product entered by the user, quantity to be restocked

```

```

        2D list containing product data

```

```

    function: it is used to update the stock by adding quantity to be restocked

```

```

    return: it returns the updated 2D list nl

```

```

    """

```

```

    nl[restock_product_id-1][2]=
                                str(int(nl[restock_product_id-1][2])+
restock_product_qty)

```

```

    return nl

```

#Bill generation in screen:

```
def restock_bill(nl, restock_product_id, restock_product_qty, purchase, total_of_all):
    """
```

parameter: 2D list nl, ID and quantity of product to be restocked, purchase list, total price of all items

function: it is used to calculate total price of each item

to add product name, quantity, unit price and total of each item to the list and

to calculate total price of all items

return: it returns the purchase list and total price of all items

```
    """
```

restock_product_name=nl[restock_product_id - 1][0] #The name of the required product is stored in restock_product_name

```
restock_unit_price=int(nl[restock_product_id- 1][3])
```

```
total_price=restock_product_qty*restock_unit_price
```

```
purchase.append([restock_product_name, restock_product_qty,
restock_unit_price, total_price])
```

```
total_of_all=total_of_all+ total_price
```

```
return purchase, total_of_all
```

6.4 main.py:

```
import datetime
```

```
from read_AakiPrajapati import read_from_file
```

```
from operations_AakiPrajapati import unique_date_time, restock_bill_items,
restock_id, restock_again, restock_qty, variables, bill_items, date_time_tdy, final_total,
buying_product_qty, buy_more, buying_product_id, buying_product_id,
enter_user_details, initialize_variable, display_stock, input_choices,
stock_display_choice1, buying_id_validation, purchase_product,
update_stock_after_purchase, shipping, stock_display_choice2, restock_id_validation,
restock_quantity_validation, stock_after_restock, restock_bill
```

```
from write_AakiPrajapati import invalid_choice, option_3, restock_bill_onscreen,  
purchase_bill_onscreen, display_free_items, bill_info, initial_shop_details, user_display,  
update_after_purchase, unique_bill, file_after_restock, bill_after_restock
```

```
#Initial display statements
```

```
initial_shop_details()
```

```
#Reading from file
```

```
nl=read_from_file()
```

```
#Displaying stock after reading from file
```

```
display_stock(nl)
```

```
#Providing the user services according to the choice they choose
```

```
#Main loop
```

```
loop=True
```

```
while loop==True:
```

```
    user_display()
```

```
    #Taking choice as input from the user
```

```
    choices=input_choices()
```

```
#Selected choice-1: Selling products to the customer
```

```
if choices==1:
```

```
    #displaying print statements
```

```
    bill_info()
```

```
    #asking user details
```

```
    name, phone_no=enter_user_details()
```

```
    #initializing variables
```

```
    selling_items, total, grand_total, shipping_cost, selling_loop= initialize_variable()
```

```
    while selling_loop==True:
```

```
        stock_display_choice1(nl)
```

```
#asking product id
buying_id=buying_product_id(nl)

#Taking the quantity required by the user
product_quantity=buying_product_qty(nl)
free_products,                                quantity_to_deduct,
product_quantity=purchase_product(product_quantity,nl,buying_id)

#Providing information to the customer regarding the Buy 3 Get 1 Free Policy
display_free_items(name,free_products)

#Updating the stock after purchase
selling_items,      total=update_stock_after_purchase      (buying_id,
product_quantity, free_products, quantity_to_deduct,selling_items, total, nl)
#Asking the customer if they want to buy any more products
selling_loop=buy_more()
#Asking the customer if they want shipping to be done
shipping_cost=shipping(shipping_cost)
#Grand total
grand_total=final_total(grand_total, total, shipping_cost)
#Date and time
date_and_time_tdy=date_time_tdy()
#Updating the txt file
update_after_purchase(nl)

#Displaying the final bill on the screen
purchase_bill_onscreen(name, phone_no, date_and_time_tdy)
#accessing through the selling_items list where all the purchase information is
stored
bill_items(selling_items, shipping_cost, grand_total)
```

```
#Saving the bill to a new .txt file with unique name
unique_bill(name, phone_no, date_and_time_tdy, shipping_cost, grand_total,
selling_items)
```

```
#Selected choice-2: Restocking products from the manufacturer
```

```
elif choices==2:
```

```
    purchase, total_of_all, purchase_loop= variables()
```

```
    while purchase_loop==True:
```

```
        stock_display_choice2(nl)
```

```
    #Asking the product ID from the user of the product they want to restock
```

```
    restock_product_id=restock_id(nl)
```

```
    #Taking quantity of required product from the user
```

```
    restock_product_qty=restock_qty(nl)
```

```
    #Updating the stock with new products(updating quantity)
```

```
    nl= stock_after_restock(restock_product_id, restock_product_qty,nl)
```

```
    #Updating the stock in file
```

```
    file_after_restock(nl)
```

```
    purchase,          total_of_all=restock_bill(nl,          restock_product_id,
restock_product_qty, purchase, total_of_all)
```

```
    #Asking user if they want to restock again
```

```
    purchase_loop=restock_again()
```

```
    date_and_time_tdy=date_time_tdy()
```

```
    #Displaying restock bill on screen
```

```
    restock_bill_onscreen(date_and_time_tdy)
```

```
    #accessing through the purchase list where all the purchase information is stored
```

```
restock_bill_items(total_of_all, purchase)

#Generation of unique bill
unique_num=unique_date_time()
bill_after_restock(unique_num, total_of_all, purchase, date_and_time_tdy)

elif choices==3:
    option_3()
    loop=False

else:
    invalid_choice(choices)
```

Bibliography

Draw.io, n.d. *draw.io*. [Online]

Available at: <https://www.drawio.com/>

[Accessed 13 05 2025].

Microsoft, n.d. *Microsoft Learn*. [Online]

Available at: <https://learn.microsoft.com/en-us/microsoft-365/cloud-storage-partner-program/online/branding>

[Accessed 09 05 2025].

Microsoft, n.d. *Microsoft Store*. [Online]

Available at: <https://apps.microsoft.com/detail/9msmlrh6lzf3?hl=en-US&gl=US>

[Accessed 13 05 2025].

Python, n.d. *Python*. [Online]

Available at: <https://www.python.org/community/logos/>

[Accessed 03 05 2025].