

24045771 Aaryan Koirala

 Islinton College,Nepal

Document Details

Submission ID

trn:oid:::3618:95807450

22 Pages

Submission Date

May 14, 2025, 12:33 PM GMT+5:45

2,072 Words

Download Date

May 14, 2025, 12:42 PM GMT+5:45

11,077 Characters

File Name

24045771 Aaryan Koirala

File Size

13.1 KB

6% Overall Similarity

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

Match Groups

-  **11** Not Cited or Quoted 5%
Matches with neither in-text citation nor quotation marks
-  **1** Missing Quotations 0%
Matches that are still very similar to source material
-  **1** Missing Citation 1%
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

- 1%  Internet sources
- 0%  Publications
- 6%  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.

Match Groups

-  11 Not Cited or Quoted 5%
Matches with neither in-text citation nor quotation marks
-  1 Missing Quotations 0%
Matches that are still very similar to source material
-  1 Missing Citation 1%
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

- 1%  Internet sources
- 0%  Publications
- 6%  Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

Rank	Source	Percentage
1	Submitted works Asia Pacific University College of Technology and Innovation (UCTI) on 2021-12-13	1%
2	Submitted works islingtoncollege on 2025-05-14	1%
3	Submitted works Botswana Accountancy College on 2021-11-18	<1%
4	Submitted works Griffith College Dublin on 2023-03-26	<1%
5	Internet harvest.usask.ca	<1%
6	Submitted works islingtoncollege on 2025-05-13	<1%
7	Submitted works Info Myanmar College on 2024-01-15	<1%
8	Submitted works London Design Engineering UTC on 2025-05-14	<1%
9	Submitted works islingtoncollege on 2025-05-13	<1%
10	Submitted works Westcliff University on 2024-07-10	<1%



8

Introduction

The main purpose of this coursework is to create a sale System for a local vendor known as WeCare that sells the beauty and skincare products. The system created is supposed to store details about products by reading product information from a text file known as Products, Storing all the data needed using proper data structures, and showing it in a user-readable format that includes: "Product Name, Brand Name, Quantity available in stock, cost price per items in rupees and country of origin of items."

Aims and Objectives

To create a system based on python that manages product details.
To apply a file-based system that reads data of products.
To store and control data of products using proper Data Structures.
To make invoices of products purchased by users, restocking and transactions details for further development.

Tools and Technologies Used

Python – programming language

Python 3 is most used high-level programming language that is used for coding for general purpose that is easily accessible on any platforms. It is versatile and easy-to-learn language. You can mix various types of operators in an one single expression and because of that it has less codes.

Ms Word

Microsoft Word i.e. Ms word is a most used word processing i.e document making software that is made by Microsoft corporation. Ms word is very easy to use and has so many features so you can make any type of documents. You can create documents with the number of pages you want. Also, Ms word helps in formatting the pages according to your choice. You can check grammars, spellings along with adding pictures, tables and so on.

5 Integrated Development and Learning Environment

IDLE is an Integrated Development Environment for python. It is a software which allows users / developers to create, design, test and debug & run python programs within one environment in an integrated terminal. It combines developer tools into close graphical user interface.

The features of IDLE's are:

Editor: To write codes with highlighting syntax and checking bugs as you write or type.

Compiler: To interpret human-language code into machine-language and execute it on different os.

Debugger: To test and debug codes.

Terminal: To interact with the machine's os.

4 Data Structures

4 Data Structures are a way of arranging data so that it can be obtained more efficiently

lying upon any situations. Data Structures are the fundamental of programming languages about which programs are built. Python allows users to learn these data structures in a simpler way than any other programming languages.

Primitive Data Structures:

7 It is a simple type of data structures used in programming. It is pre-defined data types and has fixed size and format.

Integer

It is a Numeric data type represented by the class called int. It has positive or negative numbers. It doesn't have any limits for its length in value on python.

Float

1 It is a Numeric data type which float class represents its value. It is real number with value after point.

String

Strings in python are arrays of bytes that represents Unicode characters. A character is a string of length one. It is represented by str class.

Boolean

10 Boolean data type has only two values that are: True and False. It is known by class bool.

NON-PRIMITIVE DATA STRUCTURES

Lists

3 It is a ordered sequence of information which can be accessible by index. It is denoted by []. It contains elements usually of the same data types or can also contain mixed data types. It is mutable because its elements can be changed.

List indices starts at 0 and also supports negative indexing too starting at -1 from the end. Len() function returns the length(number of items) of a list.

Operations on List:

Add

Remove

Dictionary

It stores the value like (key: pair of value). You can add any data types value and it can be duplicates but you cannot repeat keys and it should not be mutable.

Sets

It is unordered collection of items. You cannot have any duplicate elements in set but it is mutable that means you can add or remove elements after creating sets but you cannot directly change the individual elements within the set.

Tuple

It is sequence of ordered elements. You can mix element types in tuples. Tuples are immutable which means you cannot change any tuple created. It is represented with “()”

Example:

```
t = ("py", 2, 2.5)
```

Indices in tuples:

```
print(t[0]) # py
```

Algorithm

Algorithm is the step-by-step process that shows how the code runs.

Step 1: Start: Run the program

1: Call read_products() to load data from products.txt

If the file doesn't exist, display an error message and return a list that is empty.

For each line in file:

Analyze and break down into name, brand, stock, selling price & origin.

Change stock and selling price into integers.

Store the data as dictionary and append to product list. 2: Display's menu

Display Products

Sell Product

Restock Product

Exit

3: Ask User to enter a choice.

Step 2: Process's Menu

Display Products (Option 1)

Call display_products(products)

Print: Name, Brand, Stock, Price, Origin.

For each product:

Display in table format.

Sell product (Option 2)

Prompt user to:

Product name

Quantity

Buyer name

Call sell_product(products, name, quantity)

Loop through each product:

If name matches (case-insensitive):

If stock < quantity: show error and return False.

Calculate:

Free_items = quantity // 3

Total_quantity = quantity + free_items

If stock < quantity: show error and return False.

Minus total_quantity from stock.

Show msg to confirm and ree items.

Return True.

If product not found, return False.

If sell_product() returned True:

Call generate_invoice_sell(name,quantity,price,buyer)

Calculate:

Free_items = quantity//3

Total_price = quantity * price

Open invoice.txt in append mode.

Write:

Date/time

Buyer name

Product name

Quantity sold

Free items

Total items

Total price

Call write_products(products)

Open products.txt

For each product, write updated details

Else, Show "Product not found" or "Insufficient stock".

Step 3 : Restock product(option 3)

Prompt user for:

Product name to restock

Quantity

Vendor name

Call restock_product(products, name, quantity)

Call generate_invoice_stock(name, quantity, vendor)

Open invoice.txt in append mode

Write:

Date/Time

Vendor name

Product name

Quantity restocked

If restock_product() returned True:

Else, show “ product not found”..

Step 4: Exit(option 4)

Exit the loop and prompt user:

“ Do you want to restart the system? (yes/no)” If yes:

Call read_products to load data from products.txt

End Program

Display “Goodbye!” when user chooses not to restart

Flowchart

Figure 3. Flowchart

Pseudocode

Program

Main.py

This is the entrance point of the application. It deals with the general control flow of the program and integrates all the functional modules. It typically:

Shows the user (administrator), the main menu.

Receives the user input to do various actions including viewing products, recording a sale, restocking items, or exiting the program.

Calls operation.py, read.py and write.py appropriate function according to which operation was chosen.

Makes sure that the program is running in a loop for as long as the administrator will not discontinue it.

Figure 4 Main

Figure 5 main

Figure 6. main

Operation.py

Figure 7. operation

Read.py

Figure 8. read

Write.py

Figure 9. write

Testing Test case 1: To display products

Table 1. Test 1

Objective

To run the program and display all the products

Action

To display products you should enter choice: “ 1 ”

Expected Result

- Product list along with its all the details saved in the products.txt file should be displayed.

Actual Result

The products displayed successfully with all the details.

Conclusion

Test was Successful.

Evidence:

Figure 10 Test 1

Test case 2: To sell products

Table 2. Test 2

Objective

To sell the products to user.

Action

To sell products you should enter : “2”. Then, you should enter buyer’s name, product’s no & quantity of product’s.

Expected Result

After inputting all the data’s, it should print:

Product name with unit is sold

Free items given

Alert for invoice generated.

Actual Result

As expected, It run successfully.

Conclusion

Test was Successful.

Evidence:

Figure 11. Test 2

Figure 12. T2

Test case 3 : To Restock Products

Table 3. Test 3

Objective

To sell the products to user.

Action

To sell products you should enter : “2”. Then, you should enter buyer’s name, product’s no & quantity of product’s.

Expected Result

After inputting all the data’s, it should print:

Product name with unit is sold

Free items given

Alert for invoice generated.

Actual Result

As expected, It run successfully.

Conclusion

Test was Successful.

Evidence:

Figure 13. Test 3

Test case 4: To validate if you input negative value

Table 4. Test 4

Objective

To validate negative input.

Action

You should input negative value.

Expected Result

It should show “[!] Invalid choice. Please try again.” If you input negative value.

Actual Result

As expected, It showed “[!] Invalid choice. Please try again.”

Conclusion

Test was Successful.

Evidence:

Figure 14 Test 4

Test Case 5:

Table 5. Test 5

Objective

Action

Expected Result

Actual Result

Conclusion

Evidence:

Data Analysis

Skin Care Product Sale System is an inventory and sales management application that is file based and can handle product data, customer transactions and restocking vendor in an efficient manner. It works with structured data that is contained in plain text files (products.txt, and invoice.txt), and it uses modular Python scripts to moderate separate functions.

The heart of the system concerns reading product information from a file, processing sales and restocks, updating inventory and producing transaction records. Product information like name, brand, stock level, price, and origin are loaded into memory by use of read product () function which allows manipulation and display in-memory. In case, a user starts a sale, the system provides an opportunity to choose a product and its quantity applies a "Buy 3 Get 1 Free" promotional logic and defines the total

payable amount with 13% VAT. The system also ensures that there is enough stock available before sale is concluded both on the paid and the free items. If successful, then the stock is accordingly updated, the changes are pushed back to the file using write_products(), and a detailed invoice is added to invoice.txt using generate-invoice-sell().

In the same way, restocking is taken care of by validating the inputs of product selection and quantity, updating the stock in-memory, saving the changes, and recording restock activity with generate_invoice_stock().

The system makes sure that the data is stable as it ensures that the inputs are valid (e.g., positive amounts, correct product choice), and provides informative feedback in case of mistakes (e.g. with regards to inadequate stock or wrong product numbers). Any transaction is stamped and recorded, which allows traceability and fundamental historical analysis.

As a whole, the system illustrates a well-prescribed algorithm of basic inventory and transaction data processing, which will enable the effortless maintenance, transparency with invoices, and extension to such instruments as data visualization, reporting, and integration with the database.

Conclusion

Finally, the Skin Care Product Sale System has offered an overall opportunity to apply core concepts of computing, such as, file handling, modular programming, data structure implementation and user interface designing through Python. The system effectively automates product inventory management, provides support to transactional

operations, and guarantees the creation of accurate and efficient invoices in the case of sales and restocking. Integration of a pricing model that is dynamic and logic for stock adjustment i.e. 'buy three get one free offer ', the system effectively simulates how retail operations work while improving interaction with users through persistent input validation and error handling.

During the construction, the project focused on the importance of structured programming and modularity. Functions were precisely designed to encompass processes such as reading the files, validating data, showing products, issuance of invoices, and stock updates. The selection of Python dictionaries and lists as the central data structures worked favorably in structuring information for products and also facilitating effective lookups and modifications.

Testing was of great importance to validate the system's reliability. The application was tested for several products inputs and edge cases to achieve stability and correctness. Sample outputs like transaction logs and invoice files depicted the system's functionality as was expected doing this with regard to all specified requirements.

Overall, this project has broadened my knowledge on program design and implementation considerably in practical settings. It also emphasized the need for clear documentation, algorithmic planning, and good robust exception handling. Looking ahead, such hands-on experience sets a great foundation for dealing with other more complex systems in the subsequent computing exercises.

Appendix