cover

Table of Contents

[Task 01 4](#_Toc185338418)

[Task 02 8](#_Toc185338419)

[1. Global Variables Detailed Explanation 8](#_Toc185338420)

[1.1 Inventory: 8](#_Toc185338421)

[1.2 Sales History: 9](#_Toc185338422)

[2. Functions for Displaying Information 10](#_Toc185338423)

[2.1 display\_products() Function: 10](#_Toc185338424)

[2.2 display\_sales\_history() Function: 11](#_Toc185338425)

[2.3 display\_returned\_history() Function: 11](#_Toc185338426)

[2.4 display\_customer\_info() Function: 12](#_Toc185338427)

[3. Functions for Transaction Management 13](#_Toc185338428)

[3.1 sell\_product() Function: 13](#_Toc185338429)

[3.2 process\_return() Function: 14](#_Toc185338430)

[4. Main Menu and User Interaction 15](#_Toc185338431)

[**main\_menu()**. Function 16](#_Toc185338432)

[Task 03 18](#_Toc185338433)

# Task 01

A diagram of a diagram

Description automatically generated

**1. Start**

* **Purpose:** This is the entry point of the program. Every time the system runs, it begins from this point.
* **Details:** When the system is launched, it prepares to interact with the user.

**2. Display Main Menu**

* **Purpose:** Present the user with options to choose from, which are the main functionalities available in the system.
* **Details:** The system displays a list of actions that the user can select. These might include options like viewing available products, updating inventory, or checking sales histories.

**3. Get User Choice**

* **Purpose:** Allow the user to select one of the available menu options.
* **Details:** The system waits for the user to enter a number corresponding to one of the options. Based on the user’s input, the program will direct the flow to the appropriate next action.

**Decision Points (Following User Choices)**

**4. If Choice = 1 (Display Available Products)**

* **Purpose:** Show the list of products currently available in the inventory.
* **Details:**
  + When the user selects option 1, the system displays a list of products that are available for sale.
  + The products may be displayed with key details like product name, price, stock quantity, etc.

**5. If Choice = 2 (Display Returned Product Information & Refund Amount)**

* **Purpose:** Show details about products that have been returned and calculate the refund amount.
* **Details:**
  + The user is prompted to input the **Sale ID** (unique identifier for the transaction) and the **Quantity of returned products**.
  + The system will then calculate the refund amount based on the returned quantity and may also retrieve relevant information about the returned product(s) (e.g., product details, original price, refund policy).
  + After processing this information, the system will display the returned product's details and the calculated refund amount.

**6. If Choice = 3 (Update Inventory & Sales History)**

* **Purpose:** Update the inventory and sales history with new sales data.
* **Details:**
  + The user is prompted to provide **Customer Name**, **Product ID**, and **Quantity** of the items purchased.
  + The system updates:
    - The **Inventory** (subtracting the sold quantity from the stock).
    - The **Sales History** (recording the transaction, including product details, quantity sold, and customer information).
  + This ensures that the system maintains accurate records of stock and sales.

**7. If Choice = 4 (Display Sales History)**

* **Purpose:** Show the history of all sales transactions.
* **Details:**
  + When the user selects this option, the system fetches and displays all completed sales transactions.
  + This could include details like transaction ID, customer information, products sold, quantities, total amounts, etc.

**8. If Choice = 5 (Display Customer’s Transaction History)**

* **Purpose:** Display a list of transactions made by a specific customer.
* **Details:**
  + The system prompts the user to input the **Customer Name**.
  + Once provided, the system retrieves and shows a list of all transactions (sales or returns) made by that customer. This could be useful for customer support or for tracking a customer’s purchase history.

**9. If Choice = 6 (Display Return History)**

* **Purpose:** Show a history of all product returns.
* **Details:**
  + This option allows the user to see all returns that have occurred within the system.
  + It may display data such as the product ID, the quantity returned, customer name, refund amount, and the date of the return.
  + This can help track return trends and manage customer service.

**Invalid Input Handling**

**10. If Choice = 7 (End)**

* **Purpose:** End the program or session.
* **Details:**
  + If the user selects option 7, the system ends, and no further actions are taken. It’s a clean exit from the system.

**11. Invalid Choice**

* **Purpose:** Handle invalid user input.
* **Details:**
  + If the user inputs a number that doesn’t match any of the valid options (1-7), the system will display an **Invalid Choice Message**.
  + This serves as a safety net to ensure that the user is aware they made an incorrect input, prompting them to try again.

**End**

* **Purpose:** The flowchart reaches the end after the user either finishes their desired tasks or selects to exit the system (Choice 7).
* **Details:** Once the process is completed (whether the user selects the exit option or completes all tasks), the system finishes its execution.

# Task 02

## 1. Global Variables Detailed Explanation

In the inventory management system, **global variables** store all the data needed for operations like tracking products in the inventory, recording sales, and handling returns. Let’s look into each of these variables in more detail.

### 1.1 Inventory:

The **inventory** dictionary serves as the **primary data storage** for the available products in the system. Each key in the dictionary is a **product ID**, which is a unique identifier for each product. The value associated with each product ID is another dictionary containing the details of the product. Let’s break it down further:

* **product\_id (e.g., 1, 2, 3)**: The product ID is a unique integer identifier for each product. It acts as a reference key for all product-related operations (selling, returning, displaying product details).
* **Product Details Dictionary**: Each product’s dictionary contains the following fields:
  + **product\_name**: The name of the product (e.g., "Laptop", "Smartphone", etc.). This field provides a clear identification of the product.
  + **brand**: The brand name of the product (e.g., "Dell", "Samsung"). This helps in identifying the manufacturer or the company associated with the product.
  + **model**: The specific model of the product (e.g., "XPS 13", "Galaxy S21"). The model can give more details about the product’s version or variation.
  + **price**: The retail price of the product. This is a floating-point number (e.g., 999.99) that represents the selling price for the product.
  + **quantity**: The number of units available in stock. This is an integer (e.g., 10 for laptops, 15 for smartphones) that indicates how many items of a given product are available for sale in the system at any point in time.

### 1.2 Sales History:

The **sales\_history** is a **list** that keeps track of every sale that occurs in the system. Each sale is recorded as a **dictionary** with detailed information about the transaction. Let’s explore the structure of each sale record:

Each sale dictionary contains:

* **sale\_id**: A unique identifier for each sale transaction. This is generated by taking the length of the sales\_history list and adding 1. This ensures that each sale has a unique ID (e.g., sale ID 1, sale ID 2).
* **customer\_name**: The name of the customer who made the purchase. This allows the system to track which customer made which purchase.
* **product\_name**: The name of the product being sold. It helps identify the product involved in the transaction.
* **product\_id**: The ID of the product being sold. This is crucial for updating the inventory, as the system will subtract the quantity sold from the available stock for the corresponding product\_id.
* **quantity\_sold**: The number of units of the product sold in the transaction.
* **sale\_amount**: The total sale amount, calculated as the product's price multiplied by the quantity sold (e.g., $999.99 \* 2 units).
* **sale\_date**: The date of the sale. It’s stored as a string (formatted as YYYY-MM-DD) so that we can track when each transaction took place.

This **sales history** allows us to track every individual sale, giving us detailed insights into customer purchases, product popularity, and overall revenue generation.

**1.3 Returns History:**

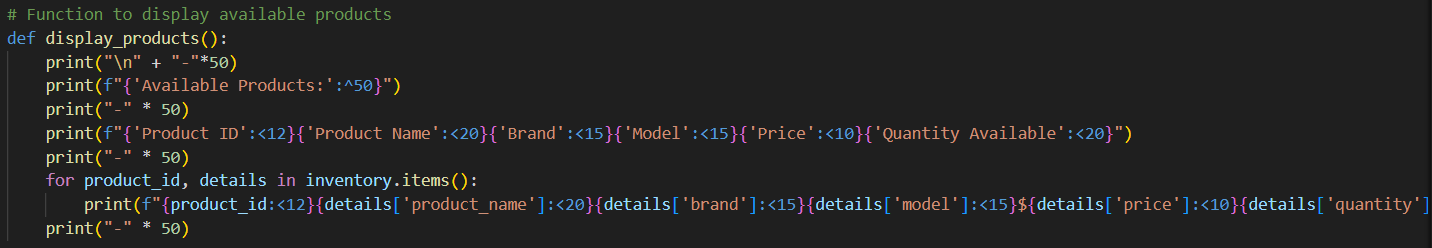
The **returns\_history** list functions similarly to the **sales\_history**, but instead of recording sales, it keeps track of returns. A return occurs when a customer decides to return a product, and this affects the inventory as well as the overall sales calculations. Each return is recorded as a dictionary containing:

* **sale\_id**: The ID of the sale that this return is associated with. This is important for matching the return to the correct sale record.
* **customer\_name**: The name of the customer who is returning the product. This helps to ensure that the return is associated with the right customer.
* **product\_name**: The name of the product being returned.
* **product\_id**: The ID of the product being returned. This will be used to update the inventory (increasing the stock of the returned product).
* **quantity\_returned**: The quantity of the product that is being returned. This is typically less than or equal to the quantity sold.
* **refund\_amount**: The amount to be refunded to the customer. This is calculated by multiplying the quantity\_returned by the product price.
* **return\_date**: The date when the return is processed. Similar to the sale date, it is stored as a string (formatted as YYYY-MM-DD).

## 2. Functions for Displaying Information

The second part of the inventory management system includes functions that display different sets of information to the user. These functions allow users to view product details, sales history, return history, and customer-specific transactions. Here's a detailed breakdown of each function in this part:

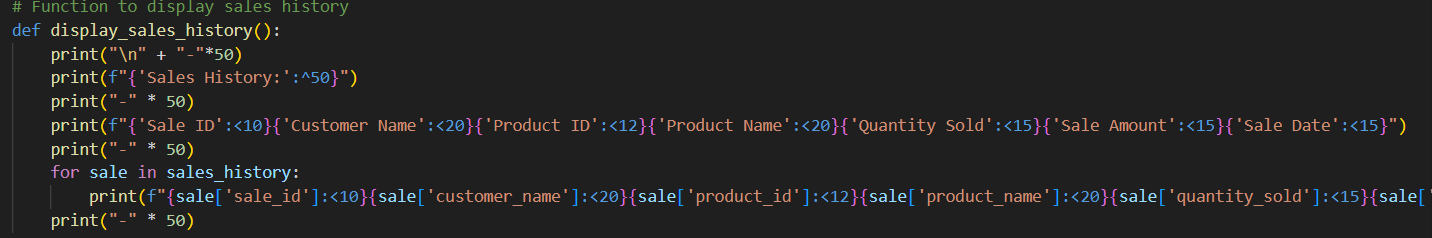
### 2.1 display\_products() Function:

****

This function displays all the products currently available in the inventory. It prints the details in a tabular format for easy readability.

* **Print Header**: The function first prints a header with the title **"Available Products"** to indicate the section.
* **Columns**: It then prints the column headers, such as **Product ID**, **Product Name**, **Brand**, **Model**, **Price**, and **Quantity Available**, formatted for alignment.
* **Product Details**: The function iterates over the inventory dictionary, printing the details of each product, including the product ID, name, brand, model, price, and quantity available.
* **Formatting**: The formatting is done using the string formatting methods to ensure that the data is displayed in an easy-to-read table format.

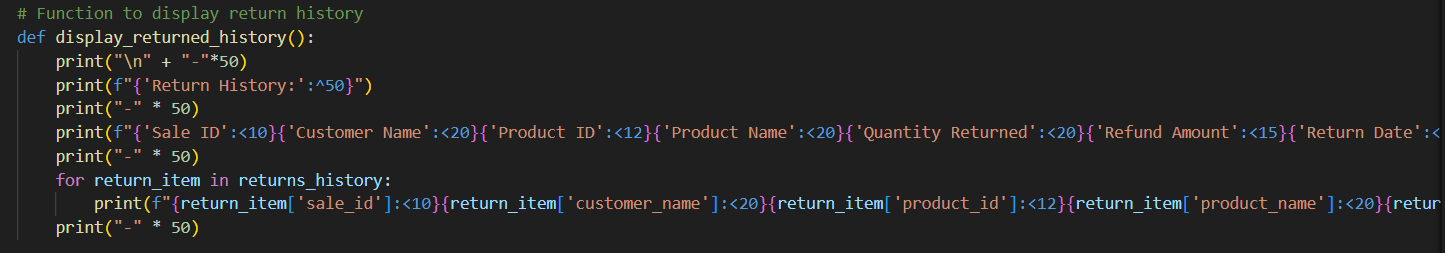
### 2.2 display\_sales\_history() Function:



This function displays the **sales history** of all completed transactions. It formats the data in a clear tabular format, showing the details of each sale.

* **Print Header**: The function prints the title **"Sales History"** to indicate that this section will show all sales made.
* **Columns**: It then prints the column headers such as **Sale ID**, **Customer Name**, **Product ID**, **Product Name**, **Quantity Sold**, **Sale Amount**, and **Sale Date**.
* **Sale Details**: It iterates through the sales\_history list and prints each sale's details, including the sale ID, customer name, product details, quantity sold, sale amount, and sale date.
* **Formatting**: As with the display\_products() function, the data is printed in a formatted table, ensuring the columns are aligned properly.

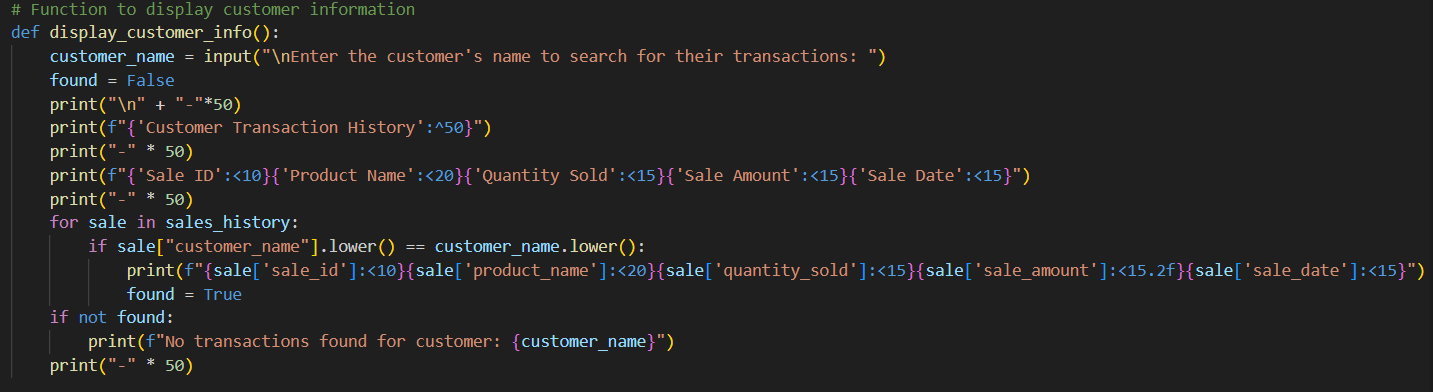
### 2.3 display\_returned\_history() Function:



This function displays the **return history** of all returned products. It prints the details of each return in a structured table.

* **Print Header**: The function begins by printing the title **"Return History"** to indicate the section.
* **Columns**: It then prints column headers such as **Sale ID**, **Customer Name**, **Product ID**, **Product Name**, **Quantity Returned**, **Refund Amount**, and **Return Date**.
* **Return Details**: The function then iterates through the returns\_history list, printing the return details for each product, including the sale ID, customer name, product details, quantity returned, refund amount, and return date.
* **Formatting**: As with the previous functions, the data is printed in a neatly formatted table for easy readability.

### 2.4 display\_customer\_info() Function:



This function allows the user to search for a customer's transaction history by entering the customer's name. It displays all the sales made by that particular customer.

* **Customer Name Input**: The function prompts the user to enter the name of the customer whose transactions they want to view.
* **Search for Transactions**: It then iterates through the sales\_history list, checking for transactions that match the entered customer name (case-insensitive).
* **Transaction Details**: For each matching sale, the function prints the sale ID, product name, quantity sold, sale amount, and sale date in a table format.
* **No Transactions Found**: If no transactions are found for the customer, it prints a message indicating that no transactions were found.

## 3. **Functions for Transaction Management**

The third part of the inventory management system includes the functions that handle key transaction processes, such as selling products and processing returns. These functions are responsible for managing the core activities that modify inventory and maintain sales and return records.

Let's break down each of these transaction-related functions in detail:

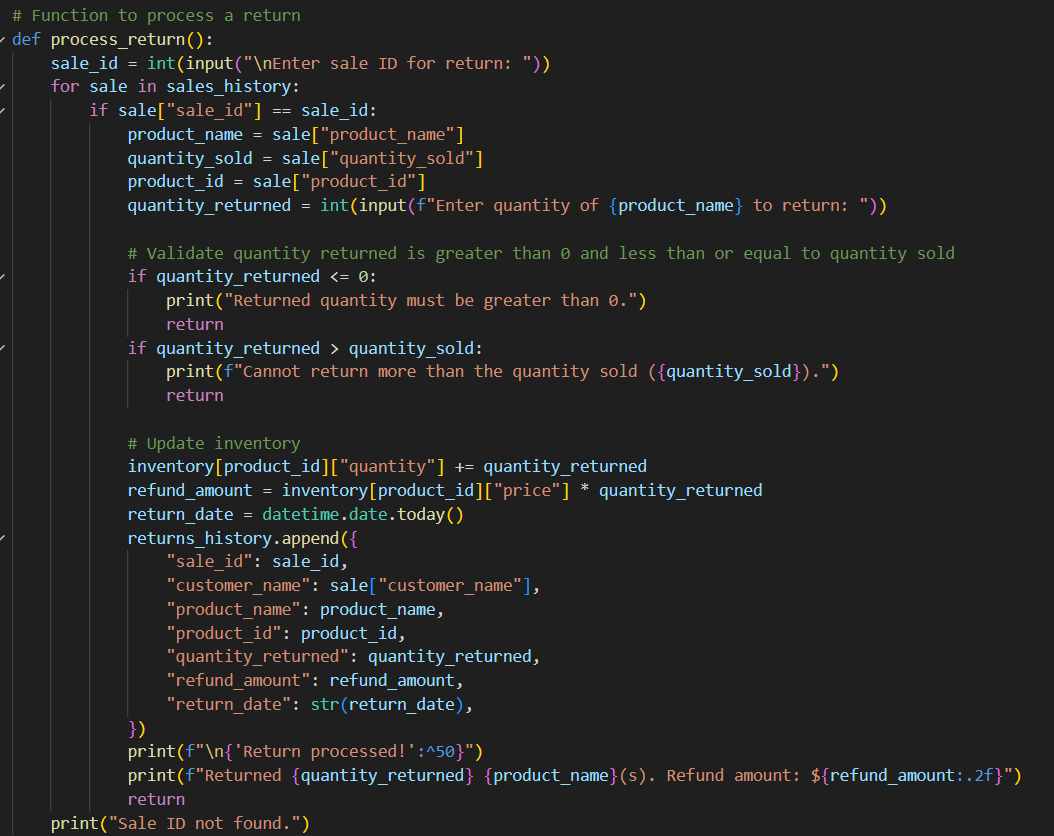
### 3.1 sell\_product() Function:



The **sell\_product()** function handles the selling of products to customers. Here's how it works step by step:

1. **Customer Input**: The function first asks for the customer's name using input().
2. **Product ID**: It then prompts the user to enter a **product ID** to select the product they want to sell.
3. **Validation of Product ID**: If the entered product ID exists in the inventory, the function proceeds. If not, it displays an "Invalid Product ID" message.
4. **Quantity Input**: Next, the user is asked to input how many units of the selected product they want to sell.
5. **Stock Check**: The function checks if the available quantity in the inventory is sufficient. If the quantity is less than the requested amount, it prints an error message. Otherwise, the sale proceeds.
6. **Update Inventory**: The inventory is updated by subtracting the sold quantity from the available stock.
7. **Sale Amount Calculation**: The sale amount is calculated by multiplying the product price by the quantity sold.
8. **Sale Record**: A record of the sale is created and appended to the sales\_history list. The sale includes the sale ID, customer name, product name, product ID, quantity sold, sale amount, and sale date.
9. **Completion Message**: Finally, the function prints a summary of the sale, including the remaining stock and the total sale amount.

### 3.2 process\_return() Function:



The **process\_return()** function handles the process of returning a product. Here’s how the function works:

1. **Sale ID Input**: The function asks the user to enter the **sale ID** associated with the return.
2. **Search for Sale**: It loops through the sales\_history to find the sale with the matching sale ID.
3. **Return Quantity Input**: Once the sale is found, the user is asked how many units of the product they want to return.
4. **Validation**:
   * If the quantity returned is less than or equal to the quantity sold, the return proceeds. If it's invalid (e.g., trying to return more than what was sold), an error message is displayed.
   * If the quantity returned is valid, the inventory is updated by adding the returned quantity back into stock.
5. **Refund Calculation**: The refund amount is calculated by multiplying the quantity returned by the price of the product.
6. **Return Record**: A record of the return is appended to the returns\_history list. The return includes the sale ID, customer name, product details, quantity returned, refund amount, and return date.
7. **Completion Message**: Finally, the function prints a summary of the return, including the quantity returned and the refund amount.

## 4. **Main Menu and User Interaction**

The fourth part of the inventory management system is the **Main Menu and User Interaction**. This section handles the main interface of the system, allowing the user to navigate through the different functions of the inventory management system, such as viewing available products, processing sales, processing returns, and displaying history.

### **main\_menu()**. Function

A screen shot of a computer program

Description automatically generated

The **main\_menu()** function is responsible for the overall user interaction and acts as the central controller of the system. Here's how it works:

1. **Displaying the Main Menu**:
   * The function continuously displays the main menu to the user in a loop (while True). It lists the available options that the user can select from:
     + **1**: Display Available Products
     + **2**: Sell a Product
     + **3**: Process a Return
     + **4**: Display Sales History
     + **5**: Display Customer Information
     + **6**: Display Return History
     + **7**: Exit the Program
2. **User Input**:
   * The function prompts the user to select an option by entering a number corresponding to one of the menu choices.
   * The user's input is stored in the variable choice.
3. **Handling User Choices**:
   * The function then uses a series of if-elif statements to determine which operation to perform based on the user’s input:
     + If the user chooses 1, it calls the **display\_products()** function to show all available products.
     + If the user chooses 2, it calls the **sell\_product()** function to handle a sale.
     + If the user chooses 3, it calls the **process\_return()** function to process a product return.
     + If the user chooses 4, it calls the **display\_sales\_history()** function to show all sales transactions.
     + If the user chooses 5, it calls the **display\_customer\_info()** function to search for and display a customer’s transaction history.
     + If the user chooses 6, it calls the **display\_returned\_history()** function to display the history of returns.
     + If the user chooses 7, the program will print an **"Exiting program"** message and exit the loop, effectively terminating the system.
4. **Handling Invalid Input**:
   * If the user enters an invalid option (not one of the numbers 1 through 7), the program will print the message **"Invalid choice. Please try again."** and prompt the user to try again.

# Task 03

Test Data Table for Inventory Management System

|  |  |  |  |
| --- | --- | --- | --- |
| Test Data | The Reason for Selecting this Data | The Expected Output | The Screenshot of the Actual Output |
| Selling 2 units of Laptop (ID: 1) | Test the basic sale process for a product with sufficient stock | The sale is successful, inventory updates, and remaining quantity is displayed |  |
| Selling 10 units of Smartphone (ID: 2) | Test when the quantity sold equals the stock | The sale is successful, inventory updates to zero |  |
| Selling 6 units of Tablet (ID: 3) | Test when stock is insufficient (only 5 available) | "Insufficient stock" error message |  |
| Processing return of 1 Laptop (ID: 1) for sale ID 1 | Validate return process for a product within the sold quantity limit | The return is processed, inventory updated, refund calculated |  |
| Processing returns of 5 units of Tablet (ID: 3) for sale ID 1 | Test return exceeding quantity sold (only 5 units sold) | "Cannot return more than the quantity sold" error message |  |
| Display sales history after sale of 2 units of Laptop (ID: 1) | Verify sales history updates after a transaction | Sales history shows correct sale details: product, quantity, amount |  |
| Display return history after returning 1 unit of Smartphone (ID: 2) | Check if return history is updated correctly | Return history shows product ID, quantity returned, refund amount, return date |  |
| Searching for customer (Eranda) who purchased 2 units of Laptop (ID: 1) | Test searching functionality for customer transactions | All sales for the customer are listed with correct product details |  |
| Selling 3 units of Smartphone (ID: 2) | Verify that sale for multiple units works correctly | The sale updates inventory, shows correct sale amount |  |
| Returning 1 unit of Tablet (ID: 3) | Check return process for a product with available stock | Inventory updates, refund amount displayed |  |
| Displaying available products after selling units | Ensure inventory list is updated after a sale | Available products show updated quantities |  |