# Introduction

## Introduction to project

The main aim of this project is to create an inventory Management System using python as part of the first milestone. The system is built for WeCare, a small-scale local vendor who mostly sells beauty and skin care products. This project’s main aim is to develop an application that can provide an effective way to manage shop daily bases operations, inventory management and invoice generation. While also contributing to the increase in sales, providing smooth shopping experience for customers and also addressing the store’s policy and offers.

The project focus on removing manual work by automating simple yet tedious tasks such as inventory management, invoice generation, store policy and offer implementation. By doing so, it saves so much time, reduces human error and dependency on manual work which can directly contribute to increase in sales. The decrease in manual work can benefit a lot for a small growing business like WeCare because the owner have to spend less money on labor cost and improving efficiency. The program uses robust input validation and error-handling to prevent mistakes in data entry, ensuring smooth and reliable operation.

The application is also responsible for generating unique invoice for each order with a unique id, product type, brand, quantity bought, name of the supplier, rate of the item, date of transaction and the total cost. The selling pricing shown on the customer interface should be marked up by 200% of the cost price. After each purchase the quantity of the product sold is reduced from the inventory and the inventory is updated in real time.

The other key function of this application is the ability to handle restocking. when new items are added in inventory. The program checks if the product already exists or not. If it exists it just adds quantity of product added if not it will add all the required information of the product needed for database. This proves that the program is scalable and can handle large inventory if needed. It should also keep track of what new products are being added, which products are low in stocks, what the current offers are available at the store and so on.

## Aims and objectives

## Technology used in coursework

### 1.3.1 Python

Python is a high-level, interpreted programming language. It uses multiple concepts of programming such as structured, object oriented (OOP) and functional programming. It was developed in 1980s by a Dutch programmer Guido van Rossum. Python is a simple yet versatile programming language. It is used by millions of users around the globe. The main reason for its popularity is due it its being incredibly beginner friendly with clean syntax and proper readability while remaining versatile for all kind of projects like Application development, AI and machine learning, web development and so on if you have the required knowledge and experience.

Python is also open source which means any user can access and learn it for free, along with a large and supportive community to solve common problems. Python’s versatility, easy to learn and adaptability allows programmers to debug problems and find it’s solution quickly and effectively. (teradata, 2023 )

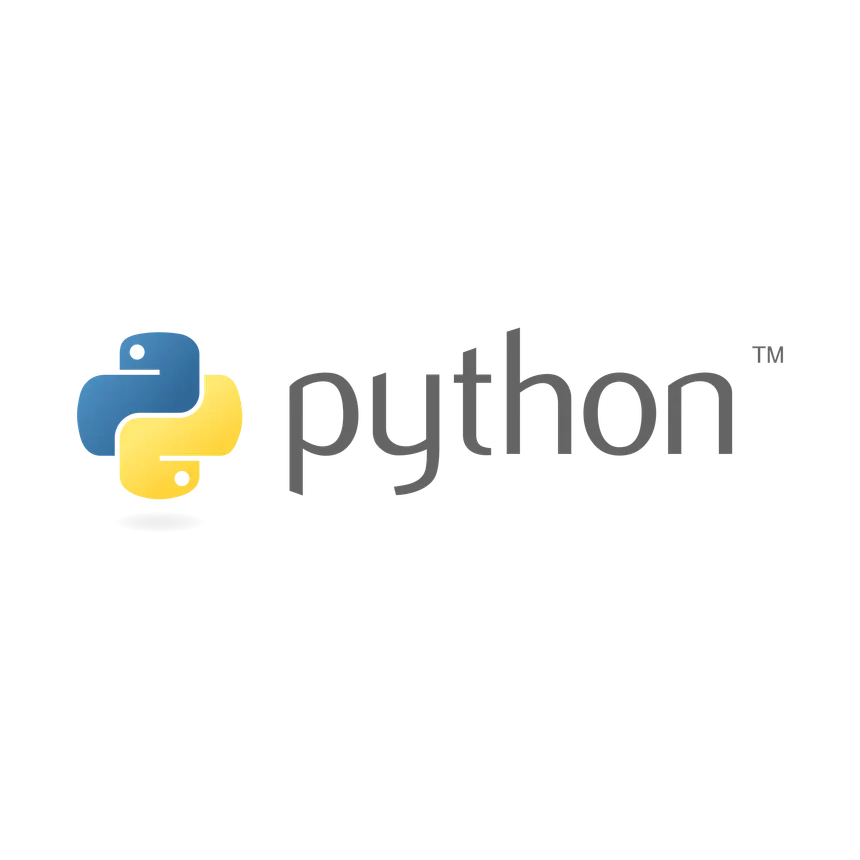


Figure 1:Python Logo

### 1.3.2 IDLE Text Editor

IDLE is Python’s self-integrated environment used for both development and learning. The main features of IDLE is that it supports cross platforms, which means it works same on MacOS, window and Linux. Its simple user interface makes it beginner friendly and easy to use. IDLE consist two main window ,the sell window and the Editor window. The sell window is used for executing commands whereas Editor window is used for writing and managing python scripts and programs

One of IDLE’s strengths is its build-in syntax highlighting, which color code for input, output, and error messages differently which makes it easier and convenient for debugging. Its simple user interface makes it beginner friendly. (Python, 2025)

# Algorithm

The step-by-step set of well-defined instructions that are designed to perform a specific task or solve a particular problem is Algorithm. It serves as the basic foundation for any computer program or system. In this program I have used algorithms to give structure to the core logic and functionality in our program like reading inventory data, checking which products are available in the system, processing sales with promotional offers like "Buy 3 Get 1 Free," restocking items, and generating invoices.

Algorithms help us break complex logic into small and easy steps, ensuring that each problem can be solved easily. By defining the algorithm for each function, we can clearly describe how the system should behave in different scenarios and can use it as a basis for converting into pseudocode and then into actual Python code.

The algorithm for the program is as follows:

## 2.1 read.py

**Algorithm for inventory\_read function :**

**Purpose**: Read and parse inventory from a text file.

**Step 1**: Start.

**Step 2**: Initialize an empty list inv.

**Step 3**: Try to open the file at filepath in read mode.

**Step 4**: For each line in the file:

* **Step 4.1**: Remove newline characters and trailing whitespace.
* **Step 4.2**: If the line is not empty:
  + **Step 4.2.1**: Split the line by commas into parts.
  + **Step 4.2.2**: If parts contains 5 elements:
    - **Step 4.2.2.1**: Convert to a dictionary with keys:
      * name, company, quantity, price, country.
    - **Step 4.2.2.2**: Append dictionary to inv.

**Step 5**: Handle FileNotFoundError:

* **Step 5.1**: Print error message.

**Step 6**: Return inv.

**Step 7**: End.

**Algorithm for check\_product function:**

**Purpose**: Check if a product exists in inventory.

**Step 1**: Start.

**Step 2**: Convert product\_name to lowercase and strip whitespace.

**Step 3**: For each item in inv:

* **Step 3.1**: If item["name"].lower() matches product\_name:
  + **Step 3.1.1**: Return the item.

**Step 4**: Return None (if not found).

**Step 5**: End.

## 2.2 write.py

**Algorithm for inventory\_add function:**

**Purpose**: Add new product details to the inventory file.

**Step 1**: Start.

**Step 2**: Open file at filepath in append mode.

**Step 3**: Write product details (name, company, quantity, price, country) in CSV format on a new line.

**Step 4**: Print "Inventory updated successfully".

**Step 5**: End.

## 2.3 Operation.py

**Algorithm for restock\_inventory function:**

**Purpose**: Restock existing or new products and update inventory.

**Step 1**: Start.

**Step 2**: Display inventory from file.

**Step 3**: Read inventory into list inv.

**Step 4**: Ask user to input supplier/company name.

**Step 5**: Initialize empty list restock\_cart.

**Step 6**: Loop while restocking:

* **Step 6.1**: Ask for product name.
* **Step 6.2**: Check if product exists using check\_product.
  + **Step 6.2.1**: If exists:
    - Ask for quantity to restock.
    - Update product quantity in inv.
    - Add to restock\_cart.
  + **Step 6.2.2**: If not found:
    - Ask if user wants to add new product.
    - If yes, collect product info using get\_from\_user.
    - Add new product to inv and restock\_cart.

**Step 7**: Ask if user wants to continue restocking.

**Step 8**: If restock\_cart is not empty:

* **Step 8.1**: Generate invoice with restocked items.
* **Step 8.2**: Save invoice as text file with a unique name.

**Step 9**: Write updated inventory back to file.

**Step 10**: End.

**Algorithm for item\_sell function:**

**Purpose**: Sell products and apply Buy 3 Get 1 Free logic.

**Step 1**: Start.

**Step 2**: Display current inventory.

**Step 3**: Read inventory into list inv.

**Step 4**: Ask for customer name.

**Step 5**: Initialize empty list item\_cart.

**Step 6**: Loop while selling:

* **Step 6.1**: Ask for product name.
* **Step 6.2**: Check if product exists.
  + If not, notify and continue loop.
* **Step 6.3**: Ask for quantity to buy.
* **Step 6.4**: Calculate free items = quantity // 3.
* **Step 6.5**: Check if stock is enough for total quantity.
  + If yes:
    - Deduct quantity + free from stock.
    - Calculate total price (quantity \* unit price).
    - Add details to item\_cart.
  + If not, notify user.

**Step 7**: Ask if user wants to buy another product.

**Step 8**: If item\_cart is not empty:

* **Step 8.1**: Generate invoice with items, free items, totals.
* **Step 8.2**: Save invoice as unique text file.
* **Step 8.3**: Write updated inventory back to file.

**Step 9**: End.

## 2.4 main.py

**Algorithm for main function:**

**Purpose**: Display menu and guide user to each operations.

**Step 1**: Start.

**Step 2**: Loop until user exits:

* **Step 2.1**: Display menu:
  1. Display Inventory
  2. Sell Product
  3. Restock Inventory
  4. Exit
* **Step 2.2**: Ask for user input.
* **Step 2.3**: Based on choice:
  1. If 1: Call inventory\_display.
  2. If 2: Call item\_sell.
  3. If 3: Call restock\_inventory.
  4. If 4: Exit loop.
  5. Else: Show "Invalid choice".

**Step 3**: End.

# Pseudocode

Pseudocode is a method of designing algorithms using plain English statements instead of using strict syntax used in programming language. In this project it is used to describe the logic and steps of a program in a way that is easy and simple for humans to understand. Even if it does not follow any specific coding standards, it shows the flow and structure of a program’s operations with enough detail to be translated into real code.

In this project, I have used pseudocode to represent the core logic behind every major function such as displaying inventory, adding new products, updating stock, and handling customer purchases. Writing pseudocode before actual programming helps in planning, identifying potential issues early, and communicating the program's logic to others who may not be familiar with the chosen programming language. It acts as a middleman between the algorithm and the actual code. I have imported pseudocode directly from the text file to keep its structure.

## Pseudocode of read.py

FUNCTION inventory\_read(filepath):

CREATE empty list inv

TRY:

OPEN file in read mode

FOR each line in file:

STRIP whitespace

IF line is not empty:

SPLIT by commas

IF exactly 5 parts:

CONVERT and CREATE product dictionary

APPEND to inv

ELSE:

PRINT format error

EXCEPT FileNotFoundError:

PRINT file not found

RETURN inv

FUNCTION inventory\_display(filepath):

CALL inventory\_read(filepath) → inv

IF inv is empty:

PRINT "No products available"

ELSE:

PRINT table header

FOR each product in inv:

PRINT product details

FUNCTION check\_product(inv, product\_name):

FOR product in inv:

IF product name matches (case-insensitive):

RETURN product

RETURN None

## Pseudocode of write.py

FUNCTION inventory\_add(filepath, product):

OPEN file in append mode

WRITE product details as CSV in line

PRINT success message

FUNCTION save\_inventory(filepath, inv):

OPEN file in write mode

FOR each product in inv:

WRITE product as CSV in line

FUNCTION write\_restock\_invoice(invoice\_file, company, restock\_cart, total\_amount):

OPEN invoice\_file in write mode

WRITE header with company name and date

FOR item in restock\_cart:

WRITE item details formatted

WRITE total amount

FUNCTION write\_sales\_invoice(invoice\_file, customer\_name, item\_cart, total\_amount):

OPEN invoice\_file in write mode

WRITE header with date and customer

FOR item in item\_cart:

WRITE item details formatted

WRITE total amount

## Pseudocode for Operation.py

FUNCTION get\_from\_user():

TRY:

GET inputs: name, company, quantity, price, origin

CONVERT quantity and price

RETURN product dictionary

EXCEPT ValueError:

PRINT invalid input

RETURN None

FUNCTION restock\_inventory(filepath):

DISPLAY inventory

READ inventory to inv

INIT restock\_cart

GET supplier name

WHILE True:

GET product name

CALL check\_product(inv, name)

IF product exists:

GET restock quantity

UPDATE quantity in inv

ADD entry to restock\_cart

ELSE:

ASK if want to add new product

IF yes:

CALL get\_from\_user() → product

CALL inventory\_add()

ADD to inv and restock\_cart

ASK if continue, BREAK if not

IF restock\_cart not empty:

GENERATE invoice filename

CALCULATE total

CALL write\_restock\_invoice()

CALL save\_inventory()

PRINT success

FUNCTION item\_sell(filepath):

READ inventory to inv

DISPLAY inventory

INIT item\_cart

GET customer name

WHILE True:

GET product name

CALL check\_product(inv, name)

IF not found:

ASK to try again or exit

CONTINUE or BREAK

GET quantity

CALCULATE free items and total qty

IF enough stock:

UPDATE inv

CALCULATE subtotal

ADD to item\_cart

ELSE:

PRINT not enough stock

ASK if buy more, BREAK if not

IF item\_cart not empty:

GENERATE invoice filename

CALCULATE total

CALL write\_sales\_invoice()

CALL save\_inventory()

PRINT success

## Pseudocode for main.py

FUNCTION get\_from\_user():

TRY:

GET inputs: name, company, quantity, price, origin

CONVERT quantity and price

RETURN product dictionary

EXCEPT ValueError:

PRINT invalid input

RETURN None

FUNCTION restock\_inventory(filepath):

DISPLAY inventory

READ inventory to inv

INIT restock\_cart

GET supplier name

WHILE True:

GET product name

CALL check\_product(inv, name)

IF product exists:

GET restock quantity

UPDATE quantity in inv

ADD entry to restock\_cart

ELSE:

ASK if want to add new product

IF yes:

CALL get\_from\_user() → product

CALL inventory\_add()

ADD to inv and restock\_cart

ASK if continue, BREAK if not

IF restock\_cart not empty:

GENERATE invoice filename

CALCULATE total

CALL write\_restock\_invoice()

CALL save\_inventory()

PRINT success

FUNCTION item\_sell(filepath):

READ inventory to inv

DISPLAY inventory

INIT item\_cart

GET customer name

WHILE True:

GET product name

CALL check\_product(inv, name)

IF not found:

ASK to try again or exit

CONTINUE or BREAK

GET quantity

CALCULATE free items and total qty

IF enough stock:

UPDATE inv

CALCULATE subtotal

ADD to item\_cart

ELSE:

PRINT not enough stock

ASK if buy more, BREAK if not

IF item\_cart not empty:

GENERATE invoice filename

CALCULATE total

CALL write\_sales\_invoice()

CALL save\_inventory()

PRINT success

# 4. Flowchart

A graphical representation of a process or an algorithm using specific symbols such as ovals, rectangles, diamonds, and arrows is known as flowchart. It is a visual representation of operations and decisions in a system which makes it easier to understand the control flow and logic behind the program operations.

For this project, I have used flowcharts to illustrate the workflows for different kinds of operations like product sale, inventory restock, invoice generation, and inventory display etc. By translating algorithms into flowcharts, it is easy to analyze how the program behaves in different conditions and interactions. Flowchart is not just design, but it also used as valuable documentation for debugging, optimization, and future enhancements. The flowcharts used in this coursework are listed below:

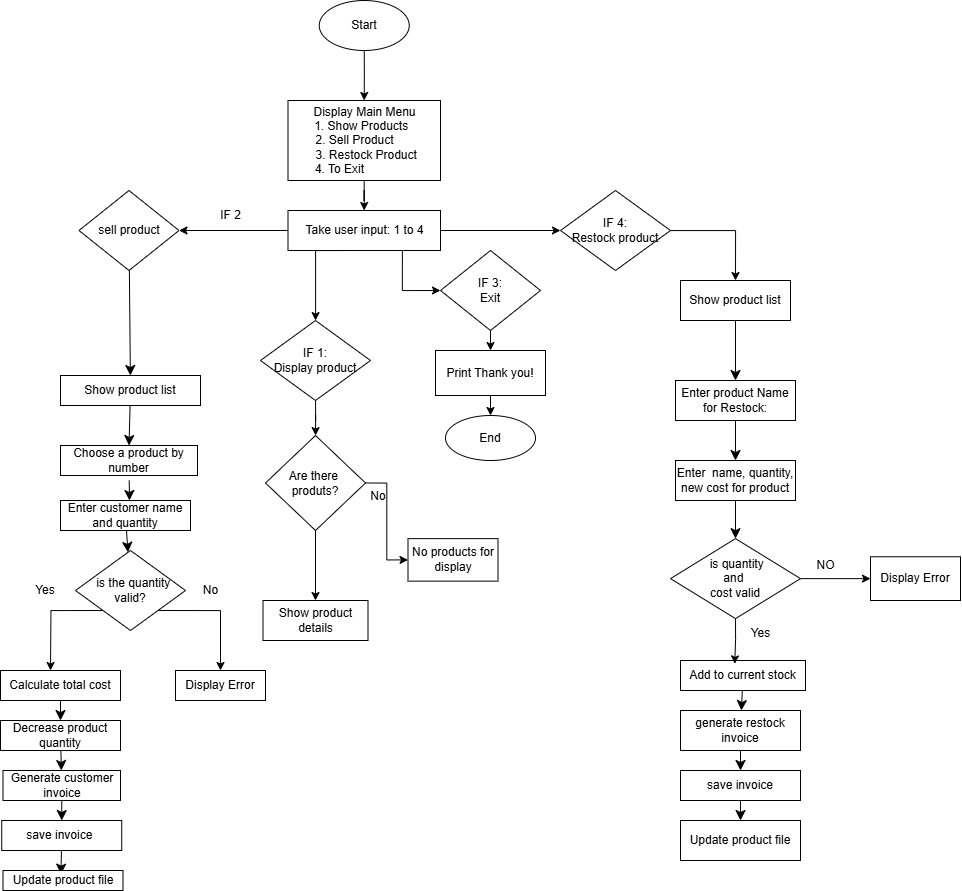


Figure 2: Flow chart

# 5. Data structure

Data structure is a data organization and storage format that is usually chosen for efficient and convenient way to manage data. Data structure provides a way to handle and manipulate data in a structured form for effective computation. In this course work I am using data structures like lists, dictionaries, strings, and set due to their simplicity and power. In this project, a combination of these data structures enables us for easy product management, efficient invoice generation, and logical promotional offer implementation. (Geeksforgeeks, 2025)

* 1. Lists

Lists contain ordered collections of related data such as product names or transaction history.

Example:

products = []

stock\_count= [200, 100, 250]

Use: For Keeping records of products details such as name, price, stock count and country of origin.

5.2 Dictionaries

Dictionaries allow for fast data retrieval with key-value pairs, enabling easy storing and retrieving of product and customer information.

Example:

product\_details = {

"name": "Vitamin C Serum",

"brand": "Garnier",

"quantity": 200,

"cost\_price": 100,

"country": "France"

}

Usage: Sorting order records by name, brand, quantity, and price for all items with a unique key.

# 6. Code Description