INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT108-3-1-PYP

PYTHON PROGRAMMING

APU1F2303CS

STUDENT NAME : **ANGELINA LEANORE**

TP NUMBER : **TP072929**

INTAKE CODE : **APU1F2303CS(IS)**

MODULE CODE : **CT108-3-1-PYP**

MODULE NAME : **PYTHON PROGRAMMING**

MODULE LECTURER : **AMARDEEP SINGH A/L UTTAM SINGH**

HAND OUT DATE : **10th MAY 2023**

HAND IN DATE : **5th JULY 2023**

WEIGHTAGE : **50%**

Table of Content

1.0 Introduction and Assumptions 3
1.1 Introduction 3
1.2 Assumptions 3
2.0 Design of The Program 4
2.1 Pseudocode 4
2.2 Flowcharts 14
3.0 Programming Source Code and Explanation 26
4.0 Screenshots of Sample Input/Output and Explanation 40
5.0 Conclusion 49
6.0 References 50

1.0 Introduction and Assumptions

1.1 Introduction

This programme for store inventory management was developed to provide effective inventory control. This programme includes fundamental features such as inputting new items, deleting items, stock taking, replenishment, adding new users, and searching. To ensure the program's security, login authentication has been implemented for three sorts of roles: admin, purchaser, and inventory checker. This technology allows you to get control of your inventory management and make smart purchase decisions.

1.2 Assumptions

- Each role must enter the security password to gain access to the features.
- An admin has access to all features, including the extra feature of adding new users.
- An inventory checker can only search items and take stock.
- A purchaser can access replenishment and search items.

2.0 Design of The Program

2.1 Pseudocode

```
PROGRAM StoreInventorySystem

BEGIN

PRINT ("Welcome to Angelina's store inventory system")

OPEN "inventory.txt" AS sourceFILE in READ MODE
    FOR each item in sourceFILE
        STRIP LEADING AND TRAILING SPACES FROM item
        CONVERT item INTO LIST USING "," AS A DELIMETER
    PRINT()
```

Insert New Item Function

```
FUNCTION insertnew_item()
    DECLARE new_item, new_description, new_category, new_unit AS STRING
    DECLARE new_code AS INTERGER
    DECLARE new_price, new_quantity,new_minimum AS INTERGER

    OPEN "inventory.txt" AS sourceFILE in READ MODE
        READ CONTENTS FROM sourceFILE INTO new_item (AS LIST)

        PRINT("Enter the new product's code: ")
        GET new_code
        FOR item in new_item
            IF new_code equals to STRIP LEADING AND TRAILING SPACES FROM item
            CONVERT item INTO LIST USING "," AS A DELIMETER to the code of current item
                PRINT("The code already exists!")
                RETURN

        PRINT("Enter the product's description: ")
        GET new_description
        PRINT("Enter the product's category: ")
        GET new_category
        PRINT("Enter the product's unit: ")
        GET new_unit
        PRINT("Enter the new product's price: ")
        GET new_price
        PRINT("Enter the product's price: ")
        GET new_price
        PRINT("Enter the product's quantity:")
        GET new_quantity
        PRINT("Enter the product's minimum: ")
        GET new_minimum

        APPEND (new_code,new_description,new_category,new_unit,new_price,new_quantity,new_minimum) TO new_item

        PRINT("Item code {new_code} has been added")
END FUNCTION
```

Update Item Function

```
FUNCTION update_item()
    DECLARE change, update_type AS STRING
    DECLARE update_code, update_type_check AS INTERGER

    inventory_data = an empty list
    OPEN "inventory.txt" AS sourceFILE in READ MODE
        FOR each data in sourceFILE
            STRIP LEADING AND TRAILING SPACES FROM data
            CONVERT data INTO LIST USING "," AS A DELIMETER
            APPEND element TO inventory_data

    update_type = False
    update_type_check = False

    WHILE not update_type
        PRINT("Enter the product's code you want to update: ")
        GET update_code
        FOR i in range(LENGTH of inventory_data)
            IF inventory_data index i equals to update_code
                WHILE not update_type_check
                    PRINT("Enter the product's {update_code} information type you would like to change: ")
                    GET update_type

                    IF update_type equals to "description"
                        PRINT("Enter the new product's description: ")
                        GET change
                        CAPITALIZE change and assign it to index i in inventory_data
                        PRINT("New product {update_type} has been updated")
                        update_type_check = True
                    ELIF update_type equals to "category"
                        PRINT("Enter the new product's category: ")
                        GET change
                        CAPITALIZE change and assign it to index i in inventory_data
                        PRINT("New product {update_type} has been updated")
                        update_type_check = True
                    ELIF update_type equals to "unit"
                        PRINT("Enter the new product's unit: ")
                        GET change
                        CAPITALIZE change and assign it to index i in inventory_data
                        PRINT("New product {update_type} has been updated")
                        update type check = True
                     ELIF update_type equals to "price"
                         PRINT("Enter the new product's price: ")
                         GET change
                         CAPITALIZE change and assign it to index i in inventory_data
                         PRINT("New product {update_type} has been updated")
                         update_type_check = True
                     ELIF update_type equals to "quantity"
                         PRINT("Enter the new product's quantity: ")
                         GET change
                         CAPITALIZE change and assign it to index i in inventory_data
                         PRINT("New product {update_type} has been updated")
                         update_type_check = True
                     ELIF update_type equals to "minimum"
                         PRINT("Enter the new product's minimum: ")
                         GET change
                         CAPITALIZE change and assign it to index i in inventory_data
                         PRINT("New product {update_type} has been updated")
                         update_type_check = True
                     ELSE
                         PRINT("Invalid choice!")

                    IF update_type_check
                        update_type = True

        IF not update_type_check
            PRINT("The Product's code doesn't exist! Please try again!")

        OPEN "inventory.txt" AS destFile in WRITE MODE
            FOR each item in inventory_data
                WRITE comma-separated AS STRING of the item's values to destFile
        PRINT("'inventory.txt' File has been updated!")
END FUNCTION
```

Delete Item Function

```
FUNCTION delete_item()
    DECLARE matching_code AS STRING
    DECLARE code AS INTERGER
    DECLARE inventory_data AS LIST

    inventory_data = an empty list
    OPEN "inventory.txt" AS sourceFILE in READ MODE
        FOR each data in sourceFILE
            STRIP LEADING AND TRAILING SPACES FROM data
            CONVERT data INTO LIST USING "," AS A DELIMETER
            APPEND element TO inventory_data

    WHILE True
        PRINT("Enter the product's code You want to delete: ")
        GET code
        matching_code = False

        FOR i in range(LENGTH of inventory_data)
            IF index i inventory_data equals to code
                REMOVE index i from inventory_data
                matching_code = True
                BREAK
        IF matching_code
            PRINT("Item has been deleted!")
            BREAK
        ELSE
            PRINT("The Product code doesn't exist! Please try again!")

    OPEN "inventory.txt" AS destFile in WRITE MODE
        FOR each item in inventory_data
            WRITE comma-separated AS STRING of the item's values to destFile
    PRINT("'inventory.txt' File has been updated!")
END FUNCTION
```

Stock Taking Function

```
FUNCTION stock_taking()
    DECLARE code, new_stock AS INTERGER
    DECLARE inventory_data AS LIST

    inventory_data = an empty list
    OPEN "inventory.txt" AS sourceFILE in READ MODE
        FOR each data in sourceFILE
            STRIP LEADING AND TRAILING SPACES FROM data
            CONVERT data INTO LIST USING "," AS A DELIMETER
            APPEND element TO inventory_data

    WHILE True
        PRINT("Enter the product's code You want to change: ")
        GET code

        FOR item in inventory_data
            IF index i item equals to code
                PRINT("The quantity of " + index item 1 + " is " + index item 5)
                STRING new_stock and assign it to index 5 in item
                PRINT("The item's quantity has been updated!")
                BREAK
        ELSE
            PRINT("The Product code doesn't exist! Please try again!")
            CONTINUE

        OPEN "inventory.txt" AS destFile in WRITE MODE
            FOR each item in inventory_data
                WRITE comma-separated AS STRING of the item's values to destFile
            PRINT("'inventory.txt' File has been updated!")
END FUNCTION
```

Replenish List Function

```
FUNCTION viewreplenish_list()
    DECLARE element AS LIST

    PRINT("This are the products that need to be replenish:")
    OPEN "inventory.txt" AS sourceFILE in READ MODE
        FOR data in sourceFILE:
            STRIP LEADING AND TRAILING SPACES FROM data
            IF INTERGER element index 5 is LESS THAN INTERGER element index 6:
                PRINT(element)
END FUNCTION
```

Stock Replenishment Function

```
FUNCTION stock_replenishment()
    DECLARE code, new_stock AS INTERGER
    DECLARE inventory_data AS LIST

    inventory_data = an empty list
    OPEN "inventory.txt" AS sourceFILE in READ MODE
        FOR each data in sourceFILE
            STRIP LEADING AND TRAILING SPACES FROM data
            CONVERT data INTO LIST USING "," AS A DELIMETER
            APPEND element TO inventory_data

    matching_code = False
    WHILE not matching_code
        PRINT("Enter the product's code You want to replenish: ")
        GET code

        FOR item in inventory_data
            IF index item equals to code
                PRINT("The quantity of " + element index 1 + " is " + element index 5)
                PRINT("Enter the item's quantity you want to add: ")
                GET new_stock
                STRING new_stock and assign it to index 5 in item
                PRINT("The item's quantity has been updated!")
                BREAK

        IF not matching_code
            PRINT("The Product code doesn't exist! Please try again!")

        OPEN "inventory.txt" AS destFile in WRITE MODE
            FOR each item in inventory_data
                WRITE comma-separated AS STRING of the item's values to destFile
            PRINT("'inventory.txt' File has been updated!")
END FUNCTION
```

Search Item Function

```
FUNCTION search_item()
    DECLARE category AS STRING
    DECLARE num, mincode, maxcode, minprice, maxprice, element AS INTERGER

    PRINT("1. Search items by code")
    PRINT("2. Search items by category")
    PRINT("3. Search items by price")
    PRINT("Choose the number from above feature: "))
    GET num AS INTERGER
    IF num is 1:
        PRINT("Enter the minimum code: "))
        GET mincode AS INTERGER
        PRINT("Enter the maximum code: "))
        GET maxcode AS INTERGER
        OPEN "inventory.txt" AS sourceFile in READ MODE
            matching_code = False
            FOR data in sourceFILE
                STRIP LEADING AND TRAILING SPACES FROM data
                IF INTERGER element index is between mincode and maxcode
                    PRINT(element)
                    matching_code = True
            IF not matching_code
                PRINT("No items found in this code range")
                CALL FUNCTION search_item()
```

```
    ELIF num is 2:
        PRINT("Enter the category: ")
        GET category AS CAPITALIZE
        OPEN "inventory.txt" AS sourceFILE in READ MODE
            matching_code = False
            FOR data in sourceFile
                STRIP LEADING AND TRAILING SPACES FROM data
                IF element index 2 is equal to category
                    PRINT(element)
                    matching_code = True
            IF not matching_code:
                PRINT("No items found in this category")
                CALL FUNCTION search_item()

    ELIF num is 3:
        FLOAT INPUT("Enter the minimum price: "))
        GET minprice
        FLOAT INPUT("Enter the maximum price: "))
        GET maxprice
        OPEN inventory.txt AS sourceFILE in READ MODE
            FOR data in sourceFILE
                matching_code = False
                STRIP LEADING AND TRAILING SPACES FROM data
                IF accept DECIMAL element index 4 is between mincode and maxcode
                    PRINT(element)
                    matching_code = True
            IF not matching_code:
                PRINT("No items found in this price range")
                CALL FUNCTION search_item()

    ELSE:
        PRINT("Invalid choice. Please Choose (1/2/3)")
        CALL FUNCTION search_item()
END FUNCTION
```

Add New User Function

```
FUNCTION addnew_user()
    DECLARE new_username, new_password, new_role AS STIRNG

    OPEN "inventory.txt" AS sourceFile in READ MODE
        READ CONTENTS FROM sourceFile INTO data (AS LIST)

    PRINT("Enter the new username: ")
    GET new_username AS CAPITALIZE
    PRINT("Enter the password: ")
    GET new_password AS LOWER letter
    PRINT("Choose the role (Admin/ Inventory Checker/ Purchaser): ")
    GET new_role AS CAPITALIZE

    APPEND (new_username,new_password,new_role)TO newuser
    OPEN "userdata.txt" AS destFILE in WRITE MODE
        FOR data in newuser
            WRITE STIP LEADING AND TRAILING SPACES FROM data
    PRINT("\nNew user has been added!")

    IF new_role equal "Admin"
        CALL FUNCTION admin()
    ELIF new_role equal "Inventory Checker"
        CALL FUNCTION inventory_checker()
    ELIF new_role equal "Purchaser"
        CALL FUNCTION purchaser()
    ELSE:
        PRINT("Invalid password or username")
END FUNCTION
```

Admin Function

```
FUNCTION admin()
    DECLARE admin_task AS INTERGER

        admin_task = 0
        WHILE admin_task is not equal to 9
            PRINT("Welcome")
            PRINT("You are logged in as Admin")
            PRINT("1. Insert New Item")
            PRINT("2. Update Item")
            PRINT("3. Delete Item")
            PRINT("4. Stock Taking")
            PRINT("5. View Replenish List")
            PRINT("6. Stock Replenishment")
            PRINT("7. Search Item")
            PRINT("8. Add New User")
            PRINT("9. Finished")

            PRINT("Choose the following features: "))
            GET admin_task AS INTERGER

            IF admin_task is 1:
                CALL FUNCTION insertnew_item()
            ELIF admin_task is 2:
                CALL FUNCTION update_item()
            ELIF admin_task is 3:
                CALL FUNCTION delete_item()
            ELIF admin_task is 4:
                CALL FUNCTION stock_taking()
            ELIF admin_task is 5:
                CALL FUNCTION viewreplenish_list()
            ELIF admin_task is 6:
                CALL FUNCTION stock_replenishment()
            ELIF admin_task is 7:
                CALL FUNCTION search_item()
            ELIF admin_task is 8:
                CALL FUNCTION addnew_user
            ELIF admin_task is 9:
                PRINT("End")
                OPEN "inventory.txt" AS sourceFILE in READ MODE
                    FOR item in sourceFILE
                        STRIP LEADING AND TRAILING SPACES FROM data
                        CONVERT item INTO LIST USING "," AS DELIMETER
            ELSE:
                PRINT("Invalid choice!")
END FUNCTION
```

Inventory Checker

```
FUNCTION inventory_checker()
    DECLARE inventory_checker_task AS STRING

        inventory_checker_task = 0
        WHILE inventory_checker_task is not equal to 3
            PRINT("Welcome")
            PRINT("You are logged in as Inventory Checker")
            PRINT("1. Stock Taking")
            PRINT("2. Search Item")
            PRINT("3. Finished")
            PRINT("Choose the following features: ")
            GET inventory_checker_task AS INTERGER

            IF inventory_checker_task is 1
                CALL FUNCTION stock_taking
            ELIF inventory_checker_task is 2
                CALL FUNCTION search_item
            ELIF inventory_checker_task is 3
                PRINT("End")
                OPEN inventory.txt AS sourceFILE in READ MODE
                    FOR item in sourceFILE
                        STRIP LEADING AND TRAILING SPACES FROM item
                        CONVERT item INTO LIST USING "," AS DELIMETER
            ELSE
                PRINT("Invalid choice")
END FUNCTION
```

Purchaser Function

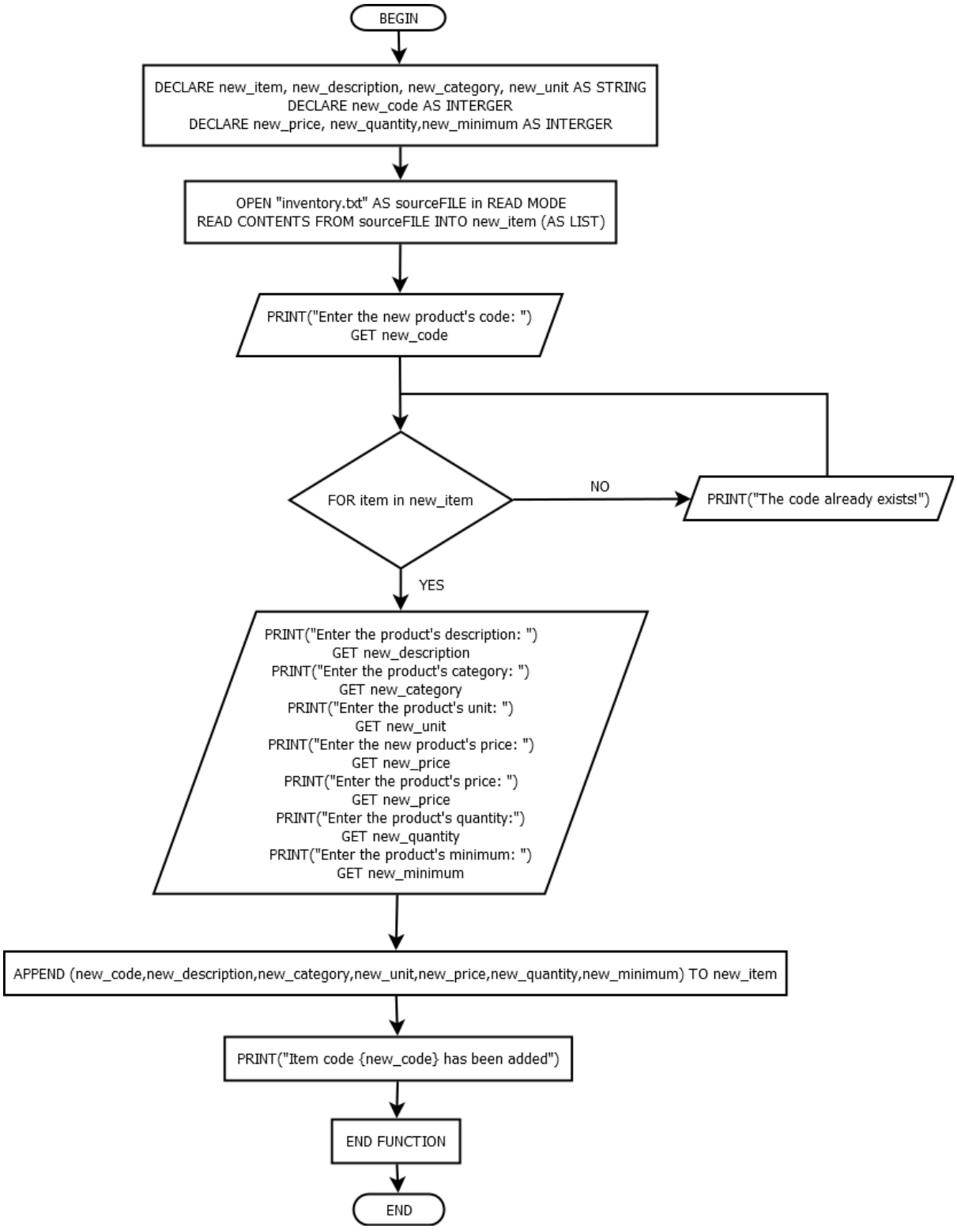
```
FUNCTION purchaser()
    DECLARE purchaser_action AS INTERGER

        purchaser_action = 0
        WHILE purchaser_action is not equal to 4
            PRINT("Welcome")
            PRINT("You are logged in as Purchaser")
            PRINT("1. View Replenish List")
            PRINT("2. Stock Replenishment")
            PRINT("3. Search Items")
            PRINT("4. End")
            PRINT("Choose the following features: ")
            GET purchaser_action AS INTERGER

            IF purchaser_action is 1:
                CALL FUNCTION viewreplenish_list()
            ELIF purchaser_action is 2:
                CALL FUNCTION stock_replenishment()
            ELIF purchaser_action is 3:
                CALL FUNCTION search_item()
            ELIF purchaser_action is 4:
                PRINT("End")
                OPEN "inventory.txt" AS sourceFILE in READ MODE
                    FOR item in sourceFILE
                        STRIP LEADING AND TRAILING SPACES FROM item
                        CONVERT item INTO LIST USING "," AS DELIMETER
            ELSE:
                PRINT("Invalid choice!")
END FUNCTION
```

Login Function

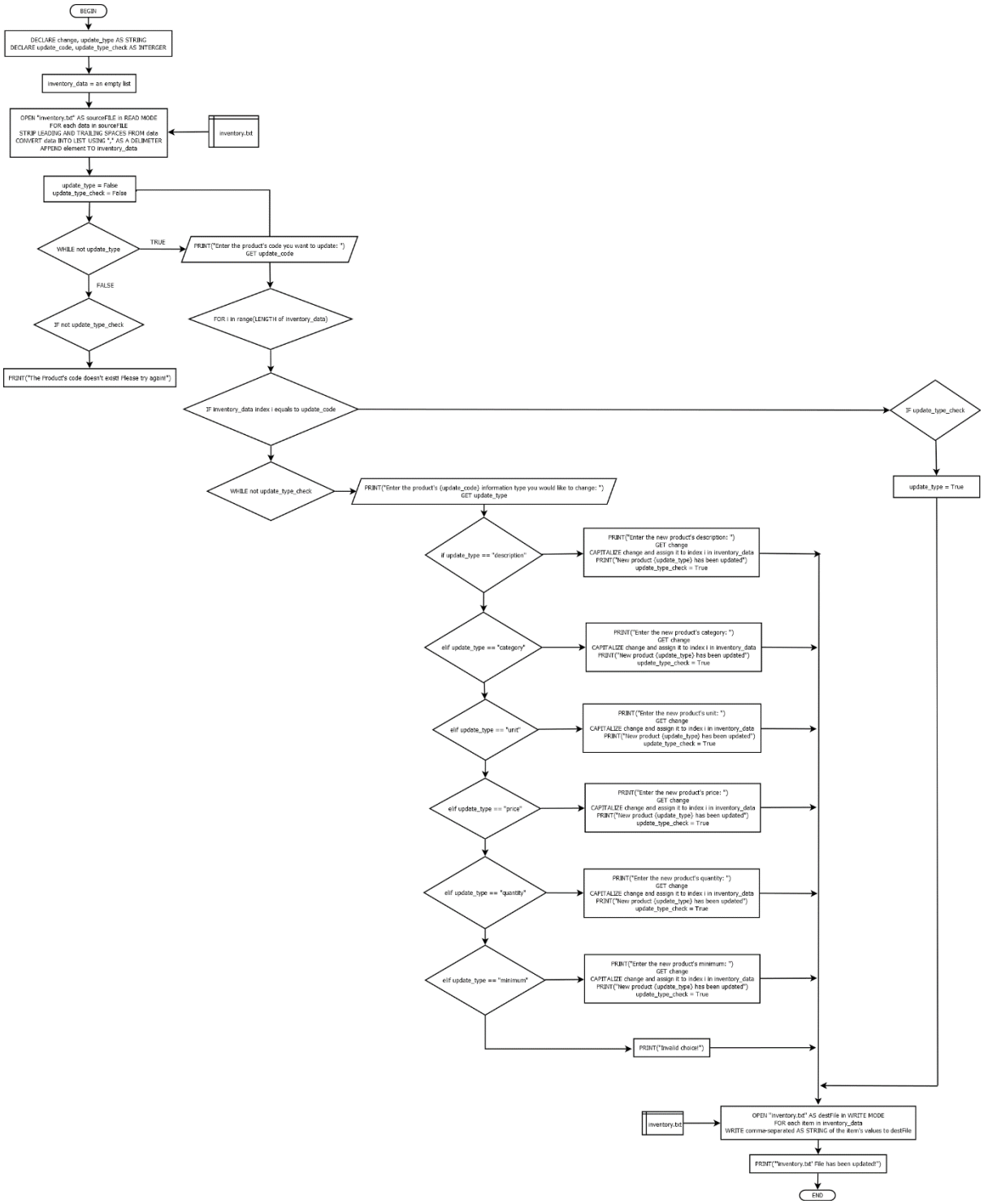
```
FUNCTION login()
    DECLARE username, password, dataFILE AS STRING

    WHILE True
        PRINT("Enter your username: ")
        GET username AS CAPITALIZE
        PRINT("Enter your password: ")
        GET password AS LOWER letter
        OPEN "inventory.txt" AS sourceFILE in READ MODE
        FOR data in sourceFILE
            STRIP LEADING AND TRAILING SPACES FROM data
            CONVERT item INTO LIST USING "," AS DELIMETER

            IF index dataFILE is equal to username and index dataFILE 1 is equal to password:
                IF index dataFILE 2 equal "Admin"
                    CALL FUNCTION admin()
                ELIF index dataFILE 2 equal "Inventory Checker"
                    CALL FUNCTION inventory_checker()
                ELIF index dataFILE 2 equal "Purchaser"
                    CALL FUNCTION purchaser()
                ELSE:
                    PRINT("Invalid role")
                RETURN
    PRINT("Invalid password or username")

END FUNCTION

CALL FUNCTION login()

END
```

2.2 Flowcharts

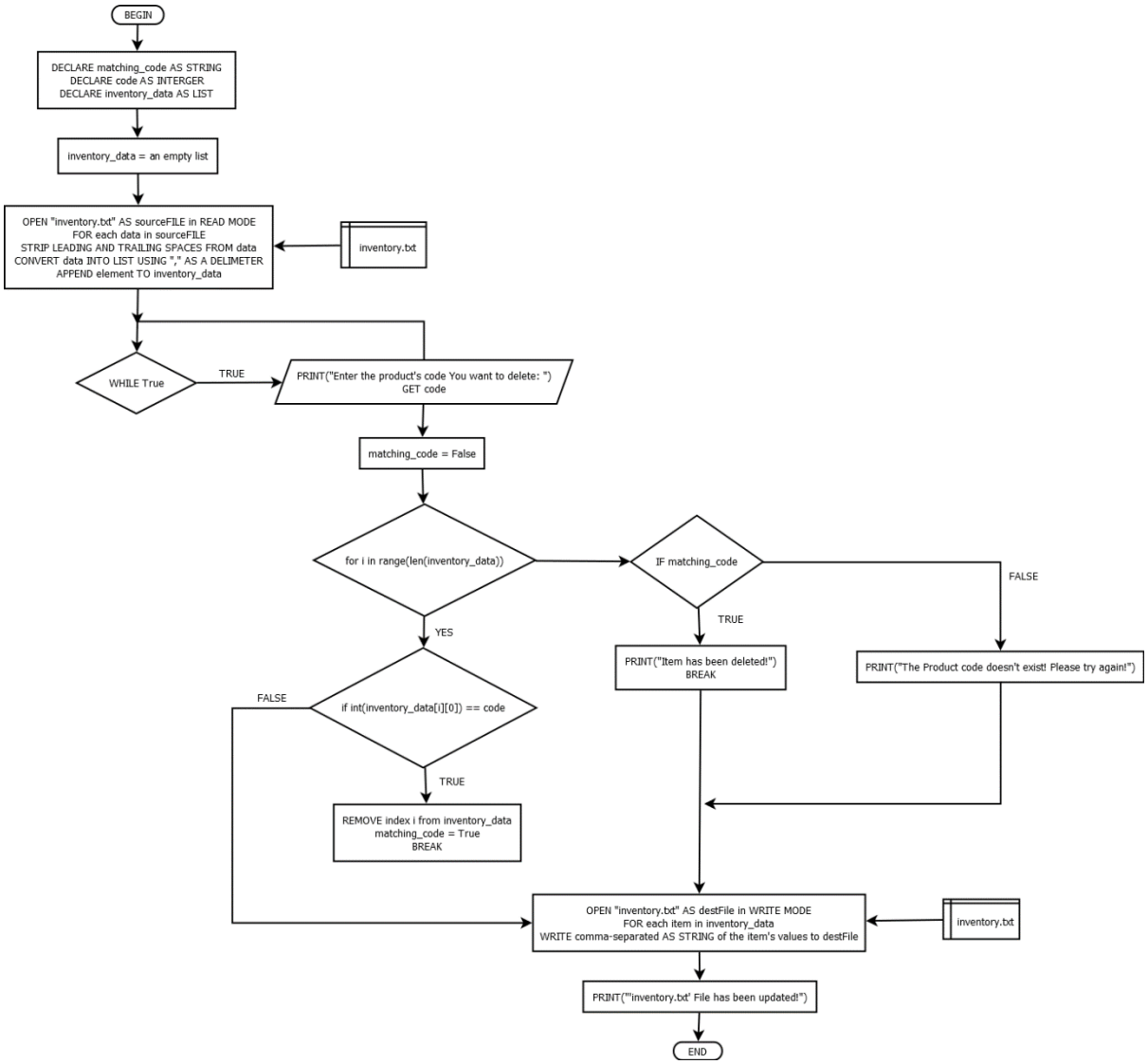
Insert New Item Function



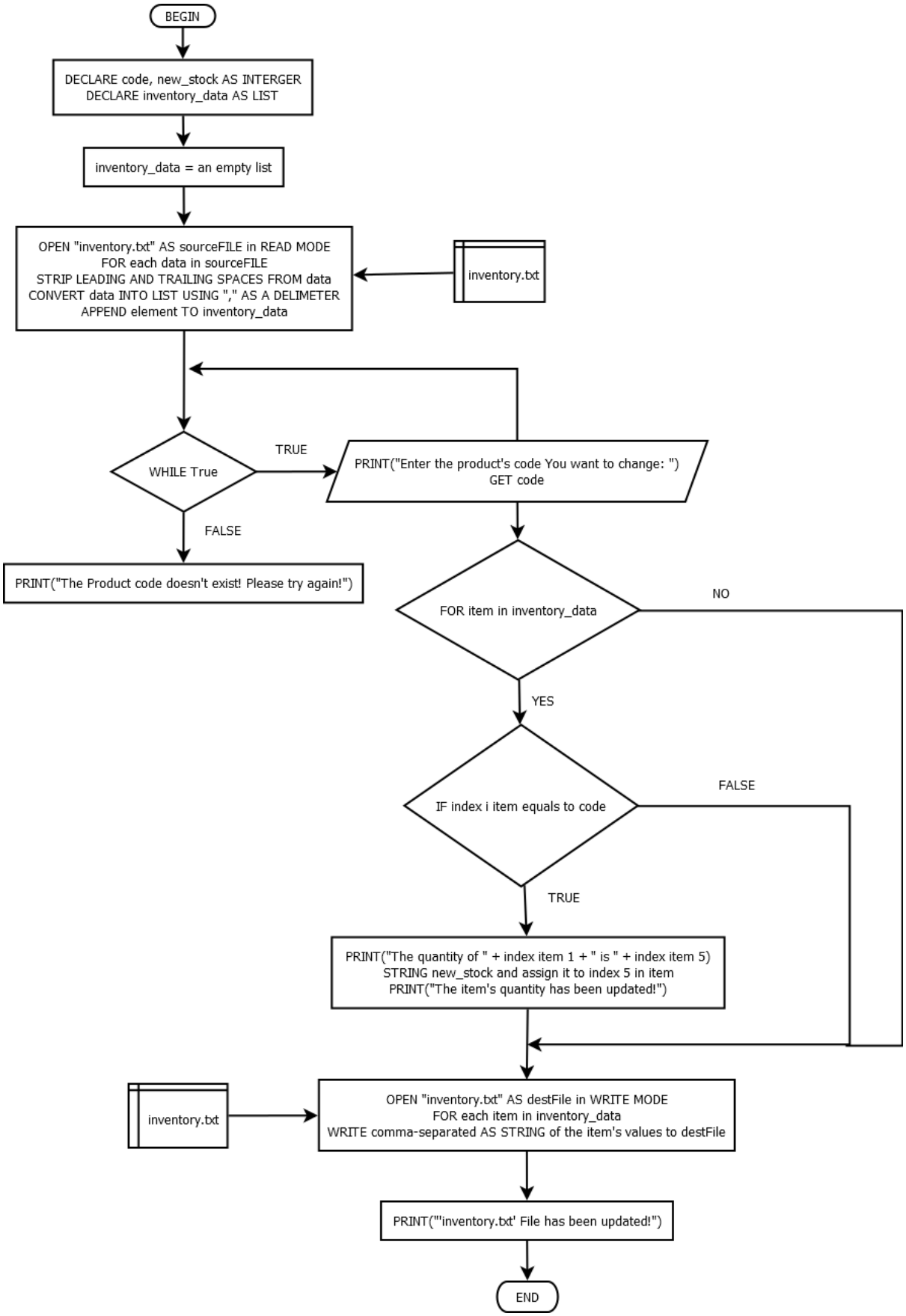
Update Item Function



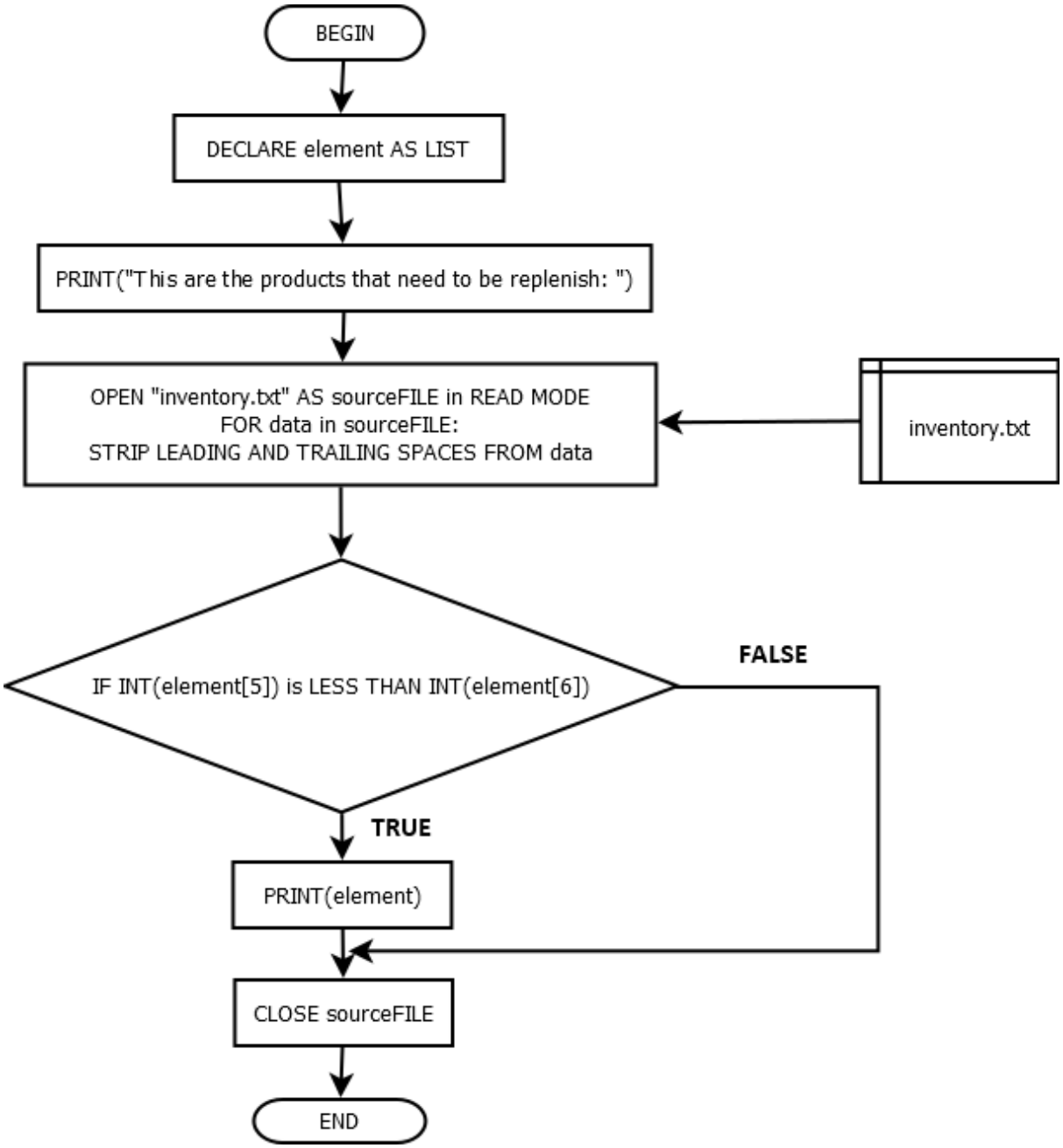
Delete Item Function



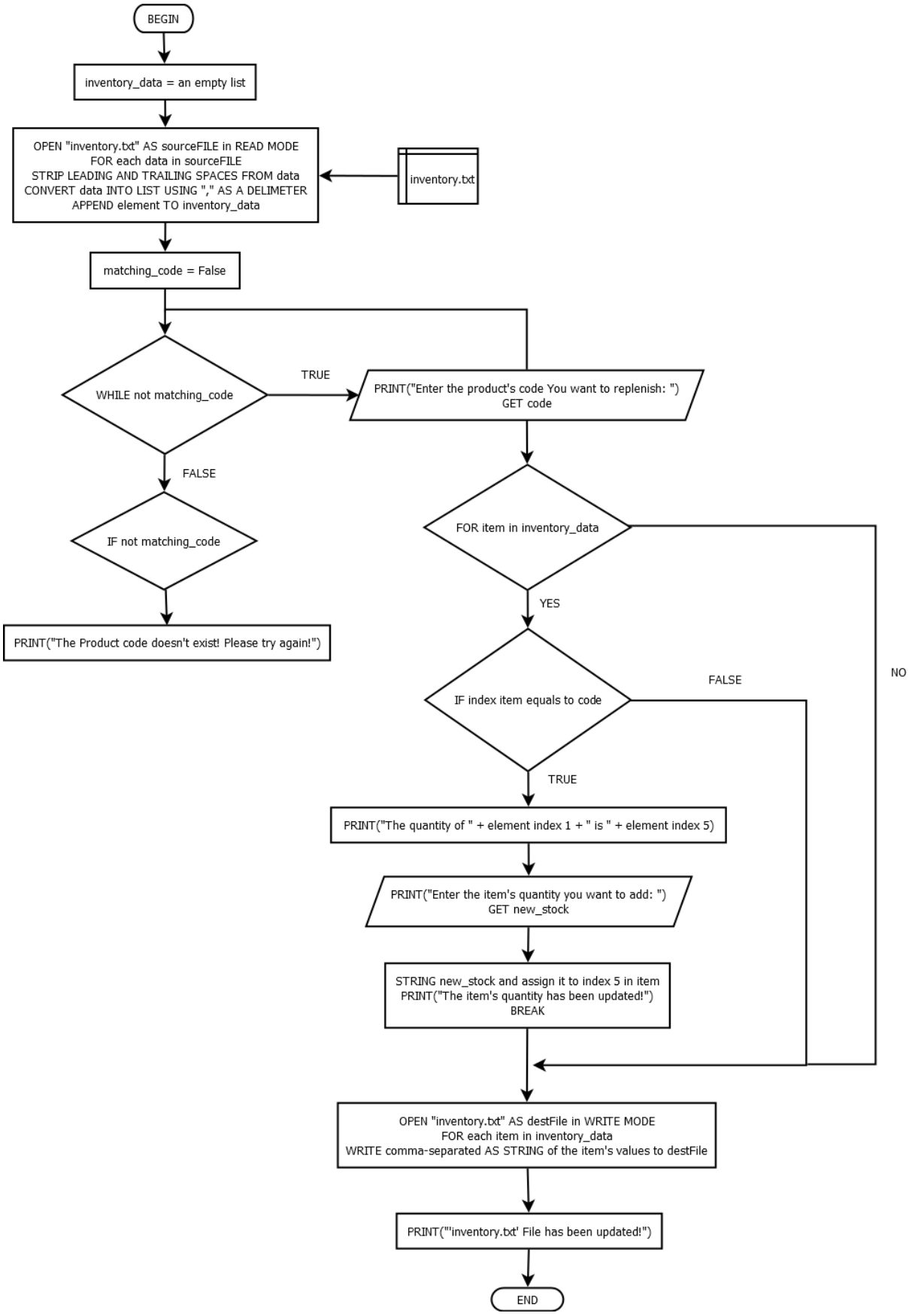
Stock Taking Function



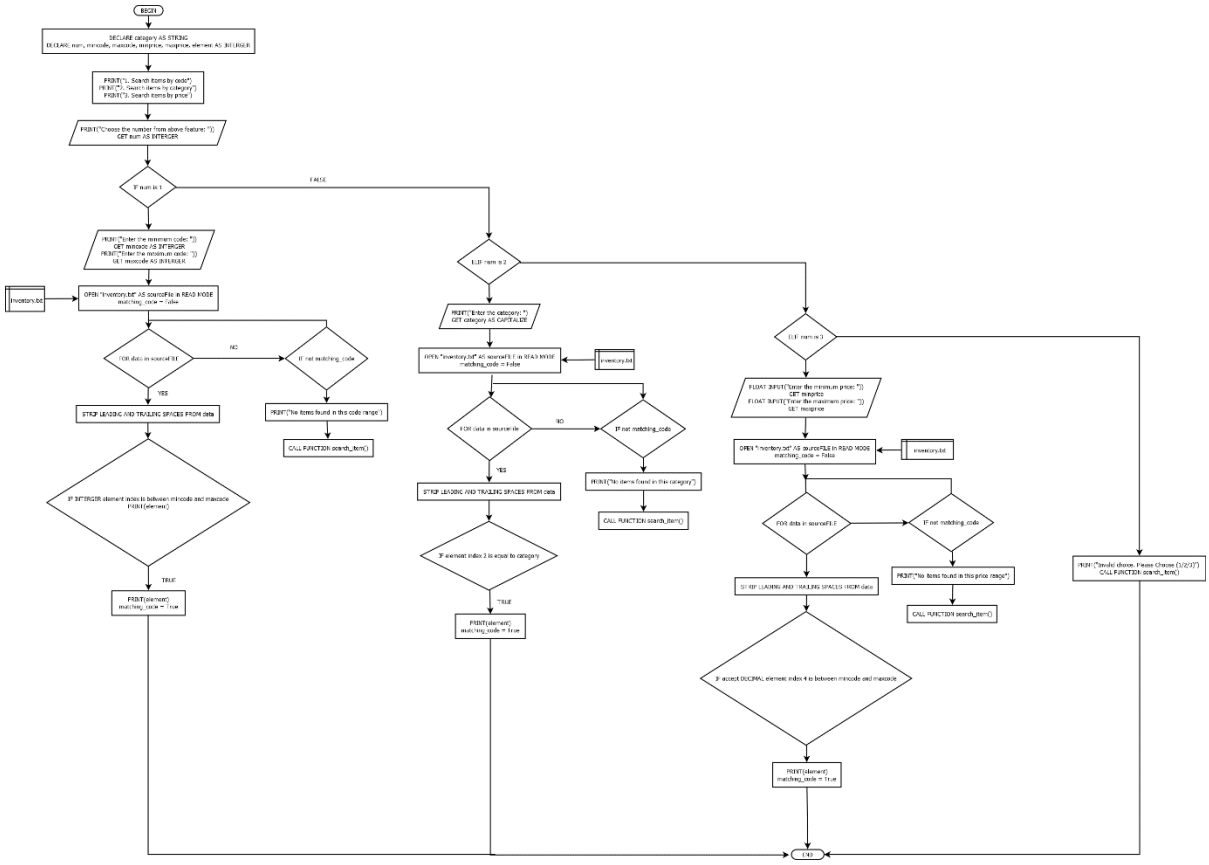
View Replenish List Function



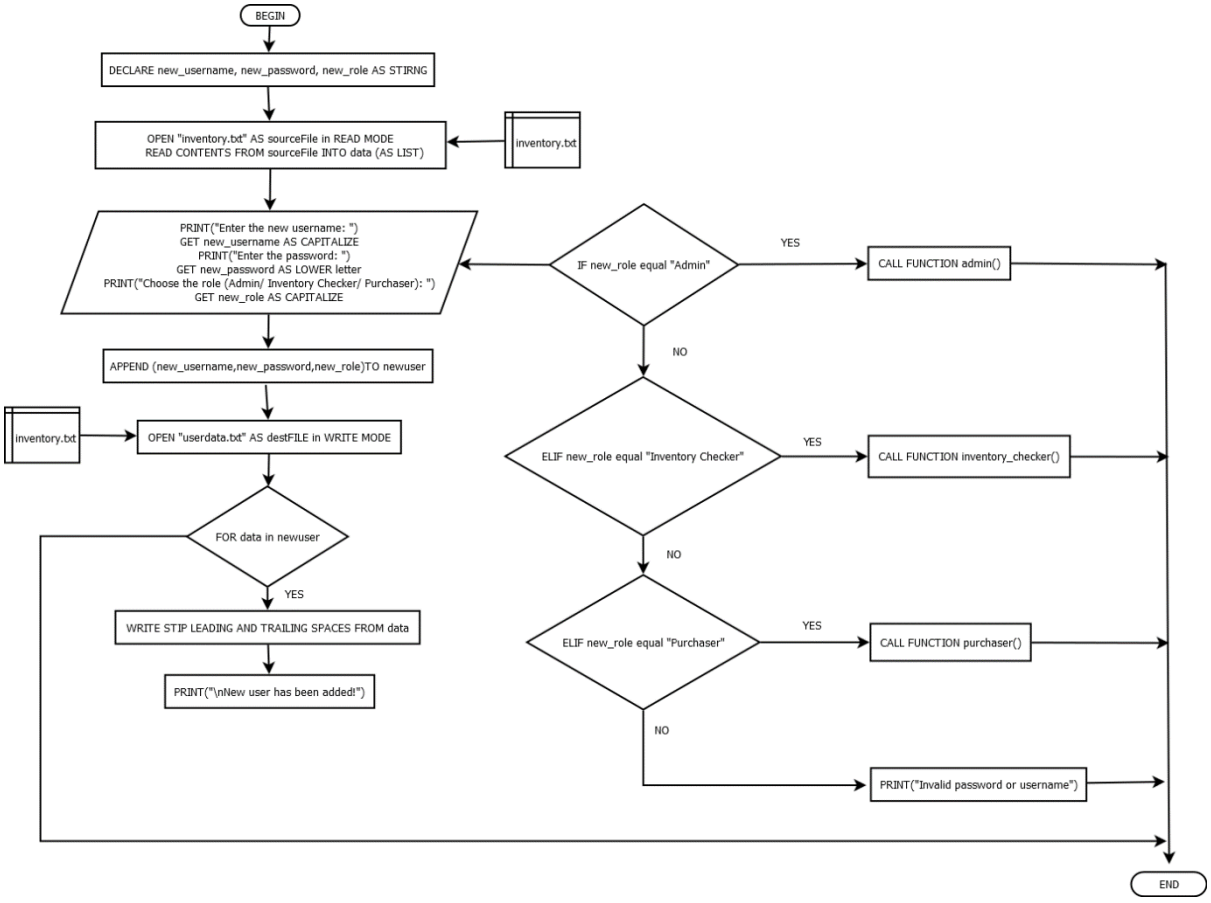
Stock Replenishment Function



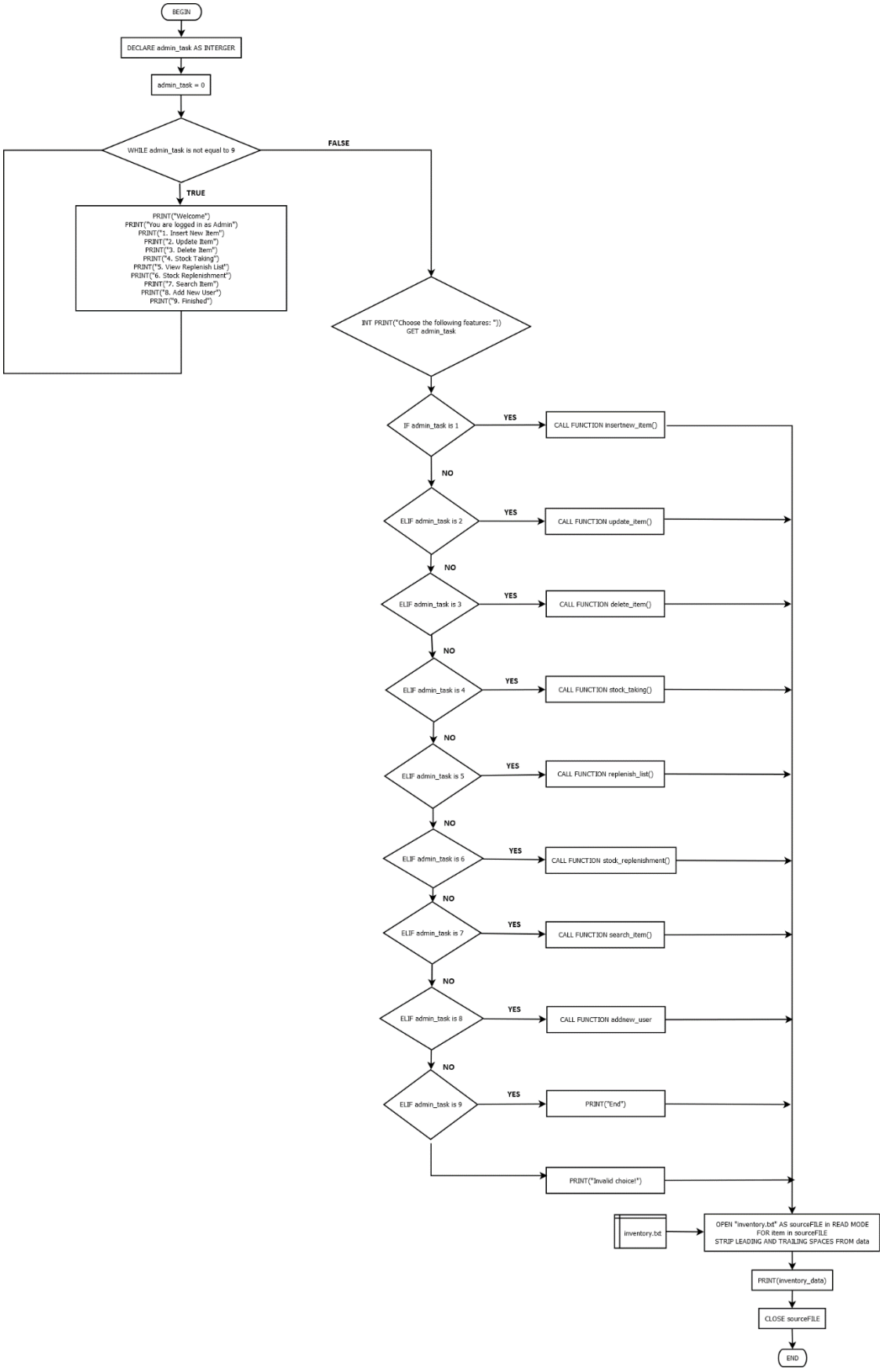
Search Item Function



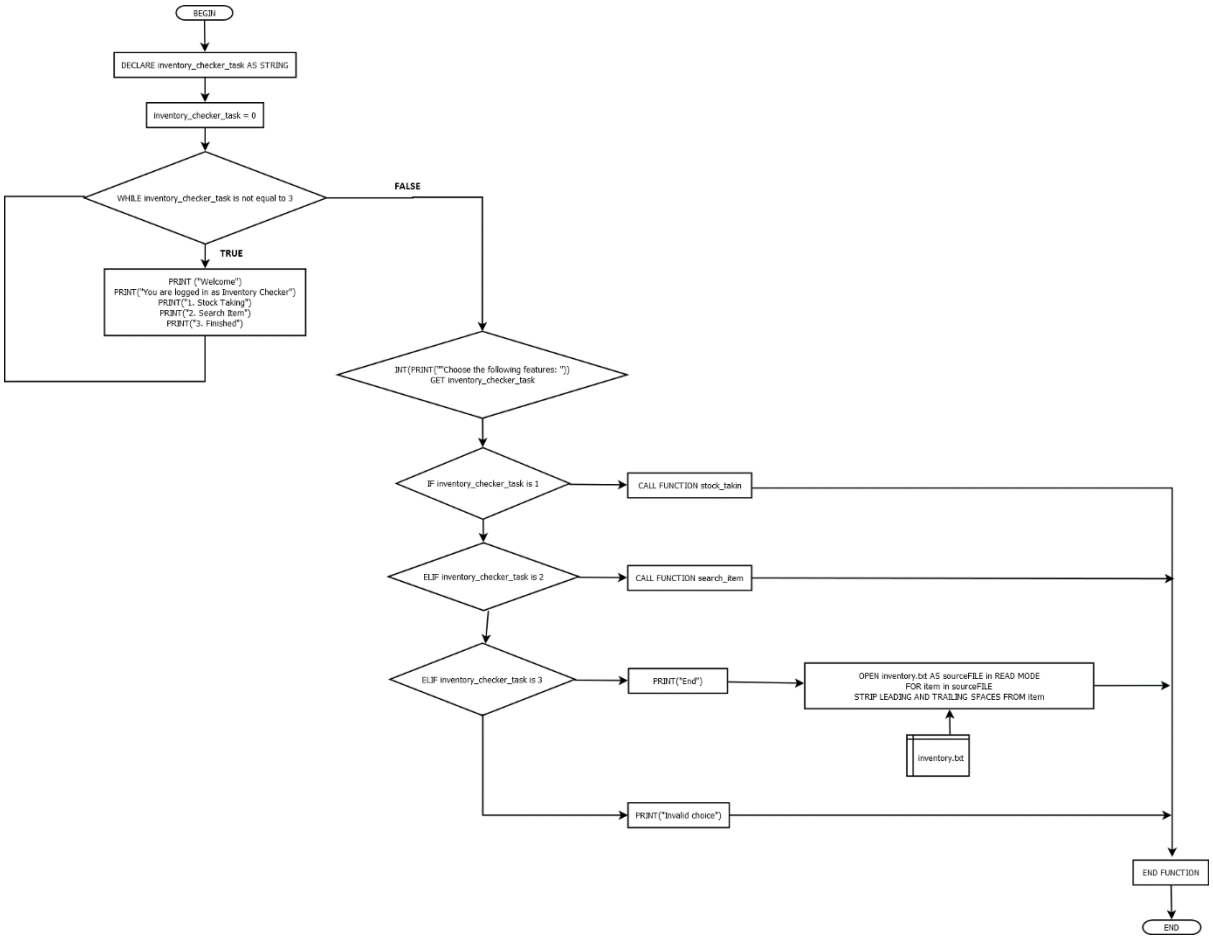
Add New User Function



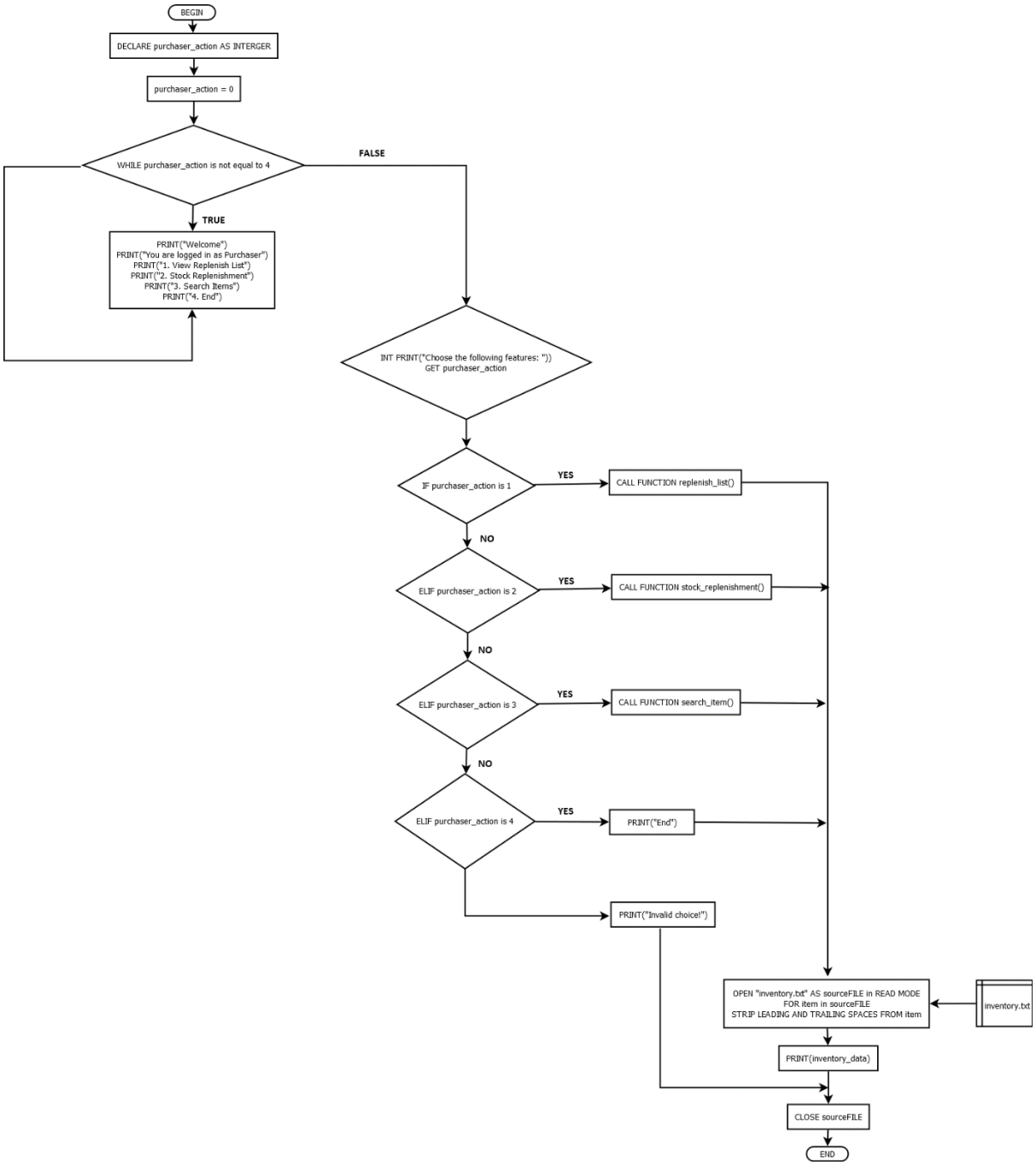
Admin Function



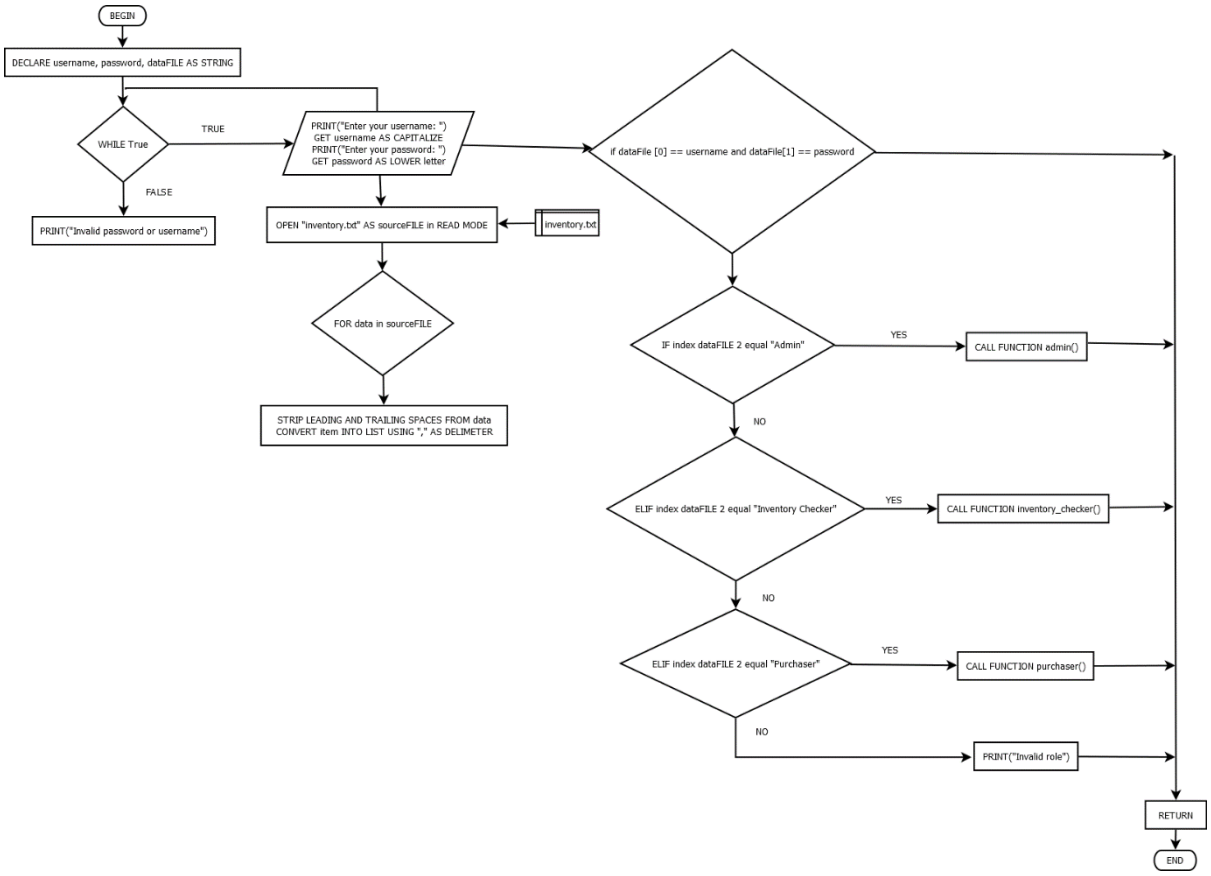
Inventory Checker Function



Purchaser Function



Login Function



3.0 Programming Source Code and Explanation

```python
#ANGELINA LEANORE
#TP072929

print("Welcome to Angelina's store inventory system")
with open('inventory.txt','r') as sourceFile:
    for item in sourceFile:
        inventory_data = item.strip().split(",")
    print()
```

This code reads the contents of the 'inventory.txt' file, strips leading and trailing spaces from each line, convert item into list using "," as a delimiter and puts the resulting data in the 'inventory_data' variable.

Insert New Item Function

```python
def insertnew_item():
    with open('inventory.txt','r') as sourceFile:
        new_item = sourceFile.readlines()

        new_code = input("Enter the new product's code: ")
        for item in new_item:
            if new_code == item.strip().split(",")[0]:
                print("\nThe code already exists!")
                return

        new_description = input("Enter the product's description: ")
        while not new_description[0].capitalize():
            print("Description must start with capital letter. Please Try Again!")
            new_description = input("Enter the product's description: ")

        new_category = input("Enter the product's category: ")
        while not new_category[0].capitalize():
            print("Category must start with capital letter. Please Try Again!")
            new_category = input("Enter the product's category: ")

        new_unit = input("Enter the product's unit: ")
        while not new_unit[0].capitalize():
            print("Unit must start with capital letter. Please Try Again!")
            new_unit = input("Enter the product's unit: ")

        new_price = float(input("Enter the product's price: "))
        new_quantity = input("Enter the product's quantity:")
        new_minimum = input("Enter the product's minimum: ")
        new_item.append("{},{},{},{},{},{},{}".format(new_code,new_description,new_
category,new_unit,new_price,new_quantity,new_minimum))

        print(f"\nItem code {new_code} has been added")
        with open('inventory.txt','w') as destFile:
            for item in new_item:
                destFile.write(item.strip() + "\n")
        print("'inventory.txt' File has been updated!")
```

This function prompts the user to add information for a new product's code after reading the file's existing 'inventory_data'. The code then continues into a "for" loop to find out if the entered "new_code" matches the existing code. It outputs the message "The code already exists!" if a match is discovered. Moreover, it will prompt you to input the form with capital letters but if its not, it will show "Description must start with capital letter. Please Try Again!". Then adds the new item to the inventory by appending it to the "inventory_data" and writing the modified "inventory.txt" back to the file.

variable	Explanation
new_item	To save the contents of the inventory.txt file
new_code	To get the new product's code
new_description	To get the new product's description
new_category	To get the new product's category
new_unit	To get the new product's unit
new_price	To get the new product's price
new quantity	To get the new product's quantity
new_minimum	To get the new product's minimum quantity
item	To cycle each sublist
float	Accept decimal
capitalize()	Changing word to capital letters

Update Item Function

```
def update_item():
    inventory_data = []
    with open("inventory.txt",'r') as sourceFile:
        for data in sourceFile.readlines():
            element = data.strip().split(",")
            inventory_data.append(element)

    update_type = False
    update_type_check = False

    while not update_type:
        update_code = input("Enter the product's code you want to update: ")
        for i in range(len(inventory_data)):
            if inventory_data[i][0] == update_code:
                while not update_type_check:
                    update_type = input(f"Enter the product's {update_code} informa
tion type you would like to change: ")
                    if update_type == "description":
                        change = input("Enter the new product's description: ")
                        inventory_data[i][1] = change.capitalize()
                        print(f"\nNew product {update_type} has been updated")
                        update_type_check = True
                    elif update_type == "category":
                        change = input("Enter the new product's category: ")
                        inventory_data[i][2] = change.capitalize()
                        print(f"\nNew product {update_type} has been updated")
                        update_type_check = True
                    elif update_type == "unit":
                        change = input("Enter the new product's unit: ")
                        inventory_data[i][3] = change.capitalize()
                        print(f"\nNew product {update_type} has been updated")
                        update_type_check = True
```

```
            elif update_type == "price":
                change = input("Enter the new product's price: ")
                inventory_data[i][4] = change
                print(f"\nNew product {update_type} has been updated")
                update_type_check = True
            elif update_type == "quantity":
                change = input("Enter the new product's quantity: ")
                inventory_data[i][5] = change
                print(f"\nNew product {update_type} has been updated")
                update_type_check = True
            elif update_type == "minimum":
                change = input("Enter the new product's minimum:")
                inventory_data[i][6] = change
                print(f"\nNew product {update_type} has been updated")
                update_type_check = True
            else:
                print("\nInvalid choice!")

        if update_type_check:
            update_type = True

if not update_type_check:
    print("The Product's code doesn't exist! Please try again!\n")

with open('inventory.txt','w') as destFile:
    for item in inventory_data:
        destFile.write(','.join([str(data) for data in item]) + '\n')
print("'inventory.txt' File has been updated!")

```

This function allows users to change product details in the ‘inventory.txt’ file. User are asked to input the product code they want to update, if the code inputted doesn’t exist it will be warning the user. Through ‘while’ loop this will keep looping until a matching input is found. Next, the user must fill in the information form that you want to replace, which will automatically be in capital letters to prevent errors. If the input is wrong, an “Invalid Choice!” will appear. The function then updates the inventory data and writes it back to the file.

Variable	Explanation
update_type	To validate the entered update_type
update_type_check	To verify the loop in the function
element	A sublist that stores information
change	To get a specified information
item	To cycle each sublist
inventory_data	Set to an empty list

Delete Function

```python
def delete_item():
    inventory_data = []
    with open ('inventory.txt', 'r') as sourceFile:
        for data in sourceFile.readlines():
            element = data.strip().split(",")
            inventory_data.append(element)

    while True:
        code = int(input("Enter the product's code You want to delete: "))
        matching_code = False

        for i in range(len(inventory_data)):
            if int(inventory_data[i][0]) == code:
                inventory_data.pop(i)
                matching_code = True
                break
        if matching_code:
            print("\nItem has been deleted!")
            break
        else:
            print("The Product code doesn't exist! Please try again!\n")

    with open('inventory.txt','w') as destFile:
        for item in inventory_data:
            destFile.write(','.join([str(data) for data in item]) + '\n')
    print("'inventory.txt' File has been updated!")

```

This function allows users to remove a product based on its code from the 'inventory.txt' file. The user is prompted to provide the deletion code in the 'code'. The programme will run continuously until a successful deletion if a 'while' loop is used. If a matching code is discovered, it will print "Item has been deleted" and the related sublist or 'element' is deleted from the 'inventory.txt' file using the 'pop()' function.

Variable	Explaination
matching_code	To check the entered code
code	To get a product's code
element	A sublist that stores information
item	To cycle each sublist
inventory_data	Set to an empty list
pop()	To removed specified information from the list
break	Exit loop

Stock Taking Function

```
def stock_taking():
    inventory_data = []
    with open ('inventory.txt','r') as sourceFile:
        for data in sourceFile.readlines():
            element = data.strip().split(",")
            inventory_data.append(element)

    while True:
        code = input("Enter the product's code you want to change: ")

        for item in inventory_data:
            if item[0] == code:
                print("The quantity of " + item[1] + " is " + item[5])
                new_stock = int(input("Enter the new item's quantity: "))
                item[5] = str(new_stock)
                print("\nThe item's quantity has been updated!")
                break
        else:
            print("The Product code doesn't exist! Please try again!\n")
            continue

        with open('inventory.txt','w') as destFile:
            for item in inventory_data:
                destFile.write(','.join([str(data) for data in item])+'\n')
        print("'inventory.txt' File has been updated!")
        break
```

This function allows users to check and update the quantity of a product. First it checks if the code entered by the user exists in the 'inventory.txt' file. If it does, the function prints "the quantity of [description] is [quantity]", if it doesn't, the function prints "The product code doesn't exist! Please try again!". Additionally, the user must enter the quantity to be entered into the quantity section of the code that has been requested.

Variable	Explanation
inventory_data	Set to an empty list
element	A sublist that stores information
code	To get a product's code
item[0]	Code
item[5]	Quantity
item	To cycle each sublist
break	Exit loop
continue	Exit loop and continue to the next iteration
new_stock	To get new item quantity

View Replenish List Function

```
def viewreplenish_list():
    print("This are the products that need to be replenish:\n")
    with open('inventory.txt', 'r') as sourceFile:
        for data in sourceFile.readlines():
            element = data.strip().split(",")
            if int(element[5]) < int(element[6]):
                print(element)

```

This function will output informayion about the items that require replenishment by reading the inventory.txt file. Evaluates the ‘quantity’ of each product with its ‘minimum’ quantity.

Variable	Explanation
element	A sublist that stores information
element[5]	Quantity
element[6]	Minimum quantity

Stock Replenishment Function

```
def stock_replenishment():
    inventory_data = []
    with open ('inventory.txt', 'r') as sourceFile:
        for data in sourceFile.readlines():
            element = data.strip().split(",")
            inventory_data.append(element)

    matching_code = False
    while not matching_code:
        code = input("Enter the product's code You want to replenish: ")

        for item in inventory_data:
            if item[0] == code:
                print("The quantity of " + item[1] + " is " + item[5])
                new_stock = int(input("Enter the item's quantity you want to add:
"))
                item[5] = str(new_stock)
                matching_code = True
                print("\nThe item's quantity has been added!")
                break

        if not matching_code:
            print("The Product code doesn't exist! Please try again!\n")

    with open('inventory.txt','w') as destFile:
        for item in inventory_data:
            destFile.write(','.join([str(data) for data in item])+'\n')
    print("'inventory.txt' File has been updated!")

```

This function allows users to add stock to a particular product in the inventory. It initially examines the inventory data before asking the user to input a product code with the prompt "Enter the product's code You want to replenish: ". Then ‘new_stock’ or quantity you want to add must be inputted. The new quantity will then be write and stored back to the ‘inventory.txt’ file.

Variable	Explaination
matching_code	To check the entered code
code	To get a product’s code
element	A sublist that stores information
item	To cycle each sublist
inventory_data	Set to an empty list
item[0]	Code
item[5]	Quantity
break	Exit loop
new_stock	To get a product quantity

Search Item Function

```
def search_item():
    print("1. Search items by code")
    print("2. Search items by category")
    print("3. Search items by price")
    num = int(input("Choose the number from above feature: "))
    if num == 1:
        mincode = int(input("Enter the product's minimum code: "))
        maxcode = int(input("Enter the product's maximum code: "))
        with open('inventory.txt','r') as sourceFile:
            matching_code = False
            for data in sourceFile.readlines():
                element = data.strip().split(",")
                if int(element[0]) >= mincode and int(element[0]) <= maxcode:
                    print(element)
                    matching_code = True
            if not matching_code:
                print("No items found in this code range\n")
                search_item()
    elif num == 2:
        category = input("Enter the product's category: ").capitalize()
        with open('inventory.txt','r') as sourceFile:
            matching_code = False
            for data in sourceFile.readlines():
                element = data.strip().split(",")
                if element[2] == category:
                    print(element)
                    matching_code = True
            if not matching_code:
                    print("No items found in this category\n")
                    search_item()
```

```
elif num == 3:
    minprice = float(input("Enter the product's minimum price: "))
    maxprice = float(input("Enter the product's maximum price: "))
    with open('inventory.txt','r') as sourceFile:
        for data in sourceFile.readlines():
            matching_code = False
            element = data.strip().split(",")
            if float(element[4]) >= minprice and float(element[4]) <= maxprice:
                print(element)
                matching_code = True
        if not matching_code:
            print("No items found in this price range\n")
            search_item()
else:
    print("Invalid choice. Please Choose (1/2/3)\n")
    search_item()
```

This function allows users to search by code, category, and price. They are prompted to choose a minimum and maximum code range if "num" is equal to 1. They are asked to submit a product 'category' if "num" is equal to 2, otherwise. They are prompted to enter a minimum and maximum price range if "num" is equal to 3. Else it will be error. The programme reads the 'inventory.txt' file.

Variable	Explanation
num	To get a chosen number
int	To hold a number
mincode	Minimal product's code
maxcode	Maximal product's code
category	To get product's category
minprice	Minimum price
maxprice	Maximum price
float	To accept decimal
element	A sublist that stores information
element[0]	Product's code
element[2]	Product's category
element[4]	Product's price
matching_code	To check the entered code
search_item()	To call search item function

Add New User Function

```
def addnew_user():
    with open("userdata.txt","r") as sourceFile:
        newuser = sourceFile.readlines()
    new_username = input("Enter the new username: ").capitalize()
    new_password = input("Enter the password: ").lower()
    new_role = input("Choose the role (Admin/ Inventory Checker/ Purchaser): ").cap
italize()
    newuser.append("{},{},{}".format(new_username,new_password,new_role))
    with open("userdata.txt","w") as destFile:
        for data in newuser:
            destFile.write(data.strip() + "\n")
    print("\nNew user has been added!")
    if new_role == "Admin":
        admin()
    elif new_role == "Inventory Checker":
        inventory_checker()
    elif new_role == "Purchaser":
        purchaser()
    else:
        print("Invalid password or username")
```

This function is only permitted for admin. It reads the 'userdata.txt' file, prompts the admin to enter the new username, password, and choose a role, which will automatically changed to capital or lower letter. Then it is appended to the user data. Afterward, this user data is written back to the file. 'admin()' is triggered if the entered "Admin". If "Inventory Checker "is chosen, 'inventory_checker()' is called. If "Purchaser" is chosen, 'purchaser()' is called. else from the following option, "invalid password or username" will be printed.

Variable	Explanation
newuser	Stores user data from userdata.txt
new_username	To get a new username
new_password	To get a new password
new_role	To get a new role
append	To add new element in the end of userdata.txt list
admin()	To call admin function
inventory_checker()	To call inventory checker function
purchaser()	To call purchaser function
capitalize()	Changing word to capital letters
lower()	Changing word to lower letters

Admin Function

```python
def admin():
    admin_task = 0
    while admin_task != 9:
        print()
        print("Welcome\nYou are logged in as Admin")
        print("1. Insert New Item")
        print("2. Update Item")
        print("3. Delete Item")
        print("4. Stock Taking")
        print("5. View Replenish List")
        print("6. Stock Replenishment")
        print("7. Search Item")
        print("8. Add New User")
        print("9. Finished")
        admin_task = int(input("Choose the following features: "))
```

```python
admin_task = int(input("Choose the following features: "))
        if admin_task == 1:
            insertnew_item()
        elif admin_task == 2:
            update_item()
        elif admin_task == 3:
            delete_item()
        elif admin_task == 4:
            stock_taking()
        elif admin_task == 5:
            viewreplenish_list()
        elif admin_task == 6:
            stock_replenishment()
        elif admin_task == 7:
            search_item()
        elif admin_task == 8:
            addnew_user()
        elif admin_task == 9:
            print("\nEnd\n")
            with open('inventory.txt','r') as sourceFile:
                for item in sourceFile:
                    inventory_data = item.strip().split(",")
        else:
            print("Invalid choice!")
```

All functionalities are available here, however only admin users are permitted to utilise this function. ‘admin_task’ will initially be set to zero. ‘insertnew_item()’ is triggered if the admin selects option 1. If 2 is choosen, 'update_item()' is called. If 3 is choosen, 'delete_item()' is called, and so on until 9 is selected which mean the 'End' program. Else from 1 to 9 option, “invalid choice!” will be printed.

Variable	Explanation
admin_task	To get choosen number
insertnew_item()	Call insert new item function
update_item()	Call update item function
delete_item()	Call delete item function
stock_taking()	Call stock taking function
viewreplenish_list()	Call view replenish list function
stock_replenishment()	Call stock replenishment function
search_item()	Call search item function
addnew_user()	Call add new user function

Inventory Checker Function

```
def inventory_checker():
    inventory_checker_task= 0
    while inventory_checker_task != 3:
        print()
        print("Welcome\nYou are logged in as Inventory Checker")
        print("1. Stock Taking")
        print("2. Search Item")
        print("3. Finished")
        inventory_checker_task = int(input("Choose the following features: "))
        if inventory_checker_task == 1:
            stock_taking()
        elif inventory_checker_task == 2:
            search_item()
        elif inventory_checker_task == 3:
            print("\nEnd\n")
            with open('inventory.txt','r') as sourceFile:
                for item in sourceFile:
                    inventory_data = item.strip().split(",")
        else:
            print("Invalid choice!")
```

This function is permitted for inventory checker role. Stock taking and search items are two of the options offered. You will prompt to choose an option you would like to choose an option you would like to run. ‘stock_taking()’ is triggered if the user selects option 1. If 2 is chosen, 'search_item()' is called. If 3 is chosen, it will end the program. else from 1 to 3 option, “invalid choice!” will be printed.

Variable	Explanation
inventory_checker_task	To get choosen number
stock_taking()	Call stock taking function
search_item()	Call search item function

Purchaser Function

```python
def purchaser():
    purchaser_action = 0
    while purchaser_action != 4:
        print()
        print("Welcome\nYou are logged in as Purchaser")
        print("1. View Replenish List")
        print("2. Stock Replenishment")
        print("3. Search Items")
        print("4. Finished")
        purchaser_action = int(input("Choose the following features: "))
        if purchaser_action == 1:
            viewreplenish_list()
        elif purchaser_action == 2:
            stock_replenishment()
        elif purchaser_action == 3:
            search_item()
        elif purchaser_action == 4:
            print("\nEnd\n")
            with open('inventory.txt','r') as sourceFile:
                for item in sourceFile:
                    inventory_data = item.strip().split(",")
        else:
            print("Invalid choice!")
```

This function is permitted for purchaser role. View replenish list, stock replenishment and search items are three of the options offered. You will prompt to choose an option you would like to choose an option you would like to run. ‘replenish_list()’ is triggered if the user selects option 1. If 2 is chosen, ‘stock_replenishment()' is called. If 3 is chosen, 'search_item()' is called. if 4 is chosen, it will end the program. else from 1 to 4 option, “invalid choice!” will be printed.

Variable	Explanation
purchase_action	To get choosen number
replenish_list()	Call replenish list function
stock_replenishment()	Call stock replenishment function
search_item()	Call search item function

Login Function

```
def login():
    while True:
        username = input("Enter your username: ").capitalize()
        password = input("Enter your password: ").lower()
        with open('userdata.txt','r') as sourceFile:
            for data in sourceFile:
                dataFile = data.strip().split(",")
                if dataFile [0] == username and dataFile[1] == password:
                    if dataFile[2] == "Admin":
                        admin()
                    elif dataFile[2] == "Inventory Checker":
                        inventory_checker()
                    elif dataFile[2] == "Purchaser":
                        purchaser()
                    else:
                        print("invalid role")
                    return
        print("Invalid password or username")
login()
```

This function in charge of granting user access to the relevant role and validating their identity based on the user's login and password. If "Admin" is typed, the function 'admin()' is called. 'inventory_checker()' is called if "Inventory Checker" is selected. 'purchaser()' is called when the option "Purchaser" is selected. "Invalid password or username" will be printed if the username and password do not match.

Variable	Explanation
username	To get a username
password	To get a password
admin()	To call admin function
inventory_checker()	To call inventory checker function
purchaser()	To call purchaser function
datafile[0]	Username
datafile[1]	Password
capitalize()	Changing word to capital letters
lower()	Changing word to lower letters

4.0 Screenshots of Sample Input/Output and Explanation

Login

```
userdata.txt
1   Leanore,mcd,Admin
2   Kevin,aw,Inventory Checker
3   Bella,kfc,Purchaser
```

As Admin

Input

```
Welcome to Angelina's store inventory system

Enter your username: leanore
Enter your password: mcd
```

Output

```
Welcome
You are logged in as Admin
1. Insert New Item
2. Update Item
3. Delete Item
4. Stock Taking
5. View Replenish List
6. Stock Replenishment
7. Search Item
8. Add New User
9. Finished
Choose the following features:
```

As Inventory Checker

Input

```
Welcome to Angelina's store inventory system

Enter your username: kevin
Enter your password: aw
```

Output

```
Welcome
You are logged in as Inventory Checker
1. Stock Taking
2. Search Item
3. Finished
Choose the following features:
```

As Purchaser

Input

```
Welcome to Angelina's store inventory system

Enter your username: bella
Enter your password: kfc
```

Output

```
Welcome
You are logged in as Purchaser
1. View Replenish List
2. Stock Replenishment
3. Search Items
4. Finished
Choose the following features:
```

```
Welcome to Angelina's store inventory system

Enter your username: gongcha
Enter your password: koi
Invalid password or username
Enter your username:
```

This is the program's login and authentication page. There are three types of role, each with username and password. The username 'Leanore' will take you to the admin page, 'Kevin' will take you to the inventory checker page. And 'Bella" will take you to the purchaser page. Based on the 'userdata.txt' file, capitalize is compulsory. This programme has been coded to automatically change the word. An error in inputting a username or password will require the user to repeat until it corrects.

Insert New Item

Input

```
Choose the following features: 1
Enter the new product's code: 1039
```

Output

```
Choose the following features: 1
Enter the new product's code: 1039

The code already exists!

Welcome
You are logged in as Admin
1. Insert New Item
2. Update Item
```

Input

```
Choose the following features: 1
Enter the new product's code: 1040
Enter the product's description: Susu
Enter the product's category: Dairy
Enter the product's unit: Box
Enter the product's price: 6
Enter the product's quantity:20
Enter the product's minimum: 10
```

Output

```
Choose the following features: 1
Enter the new product's code: 1040
Enter the product's description: Susu
Enter the product's category: Dairy
Enter the product's unit: Box
Enter the product's price: 6
Enter the product's quantity:20
Enter the product's minimum: 10

Item code 1040 has been added
'inventory.txt' File has been updated!

Welcome
You are logged in as Admin
```

If you input '1', insert new item form will appear, and you must fill up the following product's criteria. However, before that the user must input a code that is not yet in the 'inventory.txt' file, if there is, this programme will reset to the initial page, namely the admin page. After you finish filling all the requirements, the program will display 'Item code has been added', and the admin features page will remain active until it is inputted 9 'finished'. Furthermore, data in text files will automatically update what you've filled out.

Update Item

Input

```
Choose the following features: 2
Enter the product's code you want to update: 1041
```

Output

```
Choose the following features: 2
Enter the product's code you want to update: 1041
The Product's code doesn't exist! Please try again!
```

```
Enter the product's code you want to update: 1040
Enter the product's 1040 information type you would like to change: unit
Enter the new product's unit: Packs

New product unit has been updated
'inventory.txt' File has been updated!
```

If you input '2', update item form will appear. The product's code must according to the txt file, if code is incorrectly filled in then the user must fill it in again, and you are

asked to choose the product's information that will be changed. After everything done, it will show "New product unit has been updated" and change the txt file instantly.

Delete Item

Input

```
 8. Add New User
 9. Finished
 Choose the following features: 3
 Enter the product's code You want to delete: 1041
```

Output

```
 Choose the following features: 3
 Enter the product's code You want to delete: 1041
 The Product code doesn't exist! Please try again!

 Enter the product's code You want to delete:
Enter the product's code You want to delete: 1040

Item has been deleted!
'inventory.txt' File has been updated!
```

If you select 3 'Delete Item', you are required to provide the product code, which will immediately delete all of the product's information from the txt file list. Repetition will occur if an input error is made.

Stock Taking

Input

```
Choose the following features: 4
Enter the product's code you want to change: 1038
The quantity of Organic Milk is 15
Enter the new item's quantity: 0
```

Output

```
 Choose the following features: 4
 Enter the product's code you want to change: 1038
 The quantity of Organic Milk is 15
 Enter the new item's quantity: 0

 The item's quantity has been updated!
 'inventory.txt' File has been updated!
```

If you choose 4 'Stock Taking', it will show you how much stock you still have and ask you if you want to enter a new amount after you enter the code. The item's quantity has been changed in the txt file once a new quantity is entered, and the code's new quantity is finally updated.

View Replenish List

Input

```
Welcome
You are logged in as Admin
1. Insert New Item
2. Update Item
3. Delete Item
4. Stock Taking
5. View Replenish List
6. Stock Replenishment
7. Search Item
8. Add New User
9. Finished
Choose the following features: 5
```

Output

```
Choose the following features: 5
This are the products that need to be replenish:

['1038', 'Organic Milk', 'Dairy', 'Box', '5.99', '0', '2']
```

If you choose option 5, 'View Replenish List,' a list of stocks whose quantity is below the minimum will be displayed.

Stock Replenishment

Input

```
Choose the following features: 6
Enter the product's code You want to replenish: 1038
```

Output

```
Choose the following features: 6
Enter the product's code You want to replenish: 1038
The quantity of Organic Milk is 0
Enter the item's quantity you want to add: 20

The item's quantity has been added!
'inventory.txt' File has been updated!
```

If you select 6 'Stock Replenishment', it will show you how much stock you still have and you are prompt to enter the item quantity. The item's quantity will eventually be changed in the txt file by adding the item's original amount and the new quantity in the txt file.

Search Item

By Code

Input

```
Choose the following features: 7
1. Search items by code
2. Search items by category
3. Search items by price
Choose the number from above feature: 1
Enter the product's minimum code: 1000
Enter the product's maximum code: 1020
```

Output

```
Choose the following features: 7
1. Search items by code
2. Search items by category
3. Search items by price
['1014', 'Popcorn', 'Snacks', 'Bag', '2.99', '20', '4']
['1015', 'Butter', 'Dairy', 'Pack', '3.49', '12', '2']
['1016', 'Frozen Vegetables', 'Freezer', 'Pack', '2.99', '15', '3']
['1017', 'Bell Peppers', 'Vegetables', 'Pack', '4.99', '10', '2']
['1018', 'Grapefruit', 'Fruits', 'Pack', '2.99', '20', '5']
['1019', 'Potato Chips', 'Snacks', 'Pack', '1.99', '25', '3']
['1020', 'Cream Cheese', 'Dairy', 'Pack', '2.99', '14', '4']
```

Three more functions are available if you select 7 ‘Search Items’. One: ‘Search Items by Code.’ You will be requested to enter the minimum and maximum product code for which you wish to view the information.

By Category

Input

```
Choose the following features: 7
1. Search items by code
2. Search items by category
3. Search items by price
Choose the number from above feature: 2
Enter the product's category: dairy
```

Output

```
Choose the following features: 7
1. Search items by code
2. Search items by category
3. Search items by price
Choose the number from above feature: 2
Enter the product's category: dairy
['1000', 'Milk', 'Dairy', 'Pack', '4.99', '10', '2']
['1004', 'Potato Chips', 'Dairy', 'Pack', '3.99', '12', '2']
['1005', 'Yogurt', 'Dairy', 'Pack', '2.49', '20', '5']
['1010', 'Cheese', 'Dairy', 'Pack', '6.99', '12', '2']
['1015', 'Butter', 'Dairy', 'Pack', '3.49', '12', '2']
['1020', 'Cream Cheese', 'Dairy', 'Pack', '2.99', '14', '4']
['1025', 'Greek Yogurt', 'Dairy', 'Pack', '4.99', '10', '2']
['1030', 'Heavy Cream', 'Dairy', 'Box', '3.99', '25', '2']
['1038', 'Organic Milk', 'Dairy', 'Box', '5.99', '20', '2']
```

Second: ‘Search Items by Category’. The product category for which you want to view the information must be entered first. For example, Fruits, Snacks, Dairy and Vegetables.

By Price

Input

```
Choose the following features: 7
1. Search items by code
2. Search items by category
3. Search items by price
Choose the number from above feature: 3
Enter the product's minimum price: 2
Enter the product's maximum price: 5
```

Output

```
Choose the following features: 7
1. Search items by code
2. Search items by category
3. Search items by price
Choose the number from above feature: 3
Enter the product's minimum price: 2
Enter the product's maximum price: 5
['1000', 'Milk', 'Dairy', 'Pack', '4.99', '10', '2']
['1002', 'Broccoli', 'Vegetables', 'Pack', '2.99', '10', '3']
['1004', 'Potato Chips', 'Dairy', 'Pack', '3.99', '12', '2']
['1005', 'Yogurt', 'Dairy', 'Pack', '2.49', '20', '5']
['1008', 'Oranges', 'Fruits', 'Pack', '4.99', '12', '2']
['1009', 'Trail Mix', 'Snacks', 'Pack', '4.99', '14', '3']
['1012', 'Asparagus', 'Vegetables', 'Pack', '3.99', '10', '2']
['1013', 'Apples', 'Fruits', 'Pack', '3.99', '4', '3']
['1014', 'Popcorn', 'Snacks', 'Bag', '2.99', '20', '4']
['1015', 'Butter', 'Dairy', 'Pack', '3.49', '12', '2']
['1016', 'Frozen Vegetables', 'Freezer', 'Pack', '2.99', '15', '3']
['1017', 'Bell Peppers', 'Vegetables', 'Pack', '4.99', '10', '2']
['1018', 'Grapefruit', 'Fruits', 'Pack', '2.99', '20', '5']
['1020', 'Cream Cheese', 'Dairy', 'Pack', '2.99', '14', '4']
['1022', 'Tomatoes', 'Vegetables', 'Pack', '2.49', '12', '3']
['1023', 'Pears', 'Fruits', 'Pack', '3.99', '10', '2']
['1024', 'Chocolate', 'Snacks', 'Pack', '3.49', '15', '3']
['1025', 'Greek Yogurt', 'Dairy', 'Pack', '4.99', '10', '2']
['1027', 'Green Beans', 'Vegetables', 'Pack', '2.99', '15', '4']
['1028', 'Lemons', 'Fruits', 'Pack', '3.49', '10', '2']
['1029', 'Granola Bars', 'Snacks', 'Pack', '2.99', '20', '5']
['1030', 'Heavy Cream', 'Dairy', 'Box', '3.99', '25', '2']
['1031', 'Kiwi', 'Fruits', 'Pack', '2.99', '30', '5']
['1032', 'Tortilla Chips', 'Snacks', 'Pack', '5', '15', '3']
['1033', 'Melon', 'Fruits', 'Pack', '2.99', '20', '3']
['1034', 'Pineapple', 'Fruits', 'Box', '3', '15', '5']
['1035', 'Waffles', 'Snacks', 'Pack', '5', '10', '2']
['1037', 'Organic Blueberries', 'Fruits', 'Pack', '4.99', '30', '2']
['1039', 'Green Beans', 'Vegetables', 'Pack', '2.99', '10', '10']
```

Third: ‘Search Items by Price’. The minimum and maximum product prices for which you desire to examine the information must be entered.

Add New User

Input

```
Choose the following features: 8
Enter the new username: gongcha
Enter the password: koi
Choose the role (Admin/ Inventory Checker/ Purchaser): purchaser
```

Output

```
Choose the following features: 8
Enter the new username: gongcha
Enter the password: koi
Choose the role (Admin/ Inventory Checker/ Purchaser): purchaser

New user has been added!
```

userdata.txt

```
1   Leanore,mcd,Admin
2   Kevin,aw,Inventory Checker
3   Bella,kfc,Purchaser
4   Gongcha,koi,Purchaser
5
```

Input 8 for 'Add New User" will need you to create a new username and password and apply the new user for the job. Userdata.txt list will automatically capitalize its and updated with this.

Finished

Input

```
Welcome
You are logged in as Admin
1. Insert New Item
2. Update Item
3. Delete Item
4. Stock Taking
5. View Replenish List
6. Stock Replenishment
7. Search Item
8. Add New User
9. Finished
Choose the following features: 9
```

Output

```
Choose the following features: 9

End

PS D:\PYTHON PROGRAMMING LAB\SUBMISSION>
```

Lastly, input 9: 'Finished.' Since choosing this choice will end the programme, your setup is complete.

5.0 Conclusion

To sum up, this inventory system has a variety of job and role functions. The programme uses text files like ‘userdata.txt’ and ‘inventory.txt’. where ‘userdata.txt’ maintain information about username, password, and roles and ‘inventory.txt’ displays lists of grocery-related information. Additionally, this programme gives distinct users the ability to carry out particular duties in accordance with their roles.

6.0 References

shivalibhadaniya. (2023, March 21). *Python3 - if , if..else, nested if, if-elif statements*. GeeksforGeeks. https://www.geeksforgeeks.org/python3-if-if-else-nested-if-if-elif-statements/

For Loop Flowchart - A Visual Guide. (n.d.). https://www.zenflowchart.com/blog/for-loop-flowchart

Shivakumar, S. (2023, April 14). *Python validation: Types and examples of python validation*. EDUCBA. https://www.educba.com/python-validation/

Python validation - javatpoint. www.javatpoint.com. (n.d.). https://www.javatpoint.com/python-validation#:~:text=We%20can%20validate%20the%20input,the%20flag%20status%20to%20true.

Tagliaferri, L. (2021, August 20). *Understanding lists in python 3*. DigitalOcean. https://www.digitalocean.com/community/tutorials/understanding-lists-in-python-3

JoeyJoey 343k8585 gold badges688688 silver badges683683 bronze badges. (1957, August 1). *What is the difference between the float and integer data type when the size is the same?*. Stack Overflow. https://stackoverflow.com/questions/4806944/what-is-the-difference-between-the-float-and-integer-data-type-when-the-size-is

Bolton, D. (2019, January 7). *What is INT in C, C++ and C# programming?*. ThoughtCo. https://www.thoughtco.com/definition-of-int-958297#:~:text=Int%2C%20short%20for%20%22integer%2C,numeric%20variables%20holding%20whole%20numbers.