CS4051NI\CC4059NI Fundamental of Computing

60% Individual Coursework

2023-24 Summer

Student Name: Tapendra Singh

London Met ID: 23056247

College ID: NP01NT4S240057

Assignment Due Date: Friday, October 18, 2024

Assignment Submission Date: Sunday, October 13, 2024

Word Count: 242

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Tapendra Singh

Table of Contents

Introduction

According to the project specifications, the main goal of this coursework is to use Python to construct an application system for the BRJ Furniture Store. The primary aim is to acquire hands-on experience in Python programming by constructing a working system. The program will be built using the fundamental Python features—functions, lists, dictionaries, and other data types—during this study. Delivering a user-friendly solution with efficient error handling and a simplified UI is the aim. To guarantee clarity and appropriate planning, we will also develop the program's algorithm, flowchart, and pseudocode.

Objective of the Course work

This study's main goal is to teach students Python programming through the creation of a program that makes use of dictionaries, lists, functions, and other data types. Our goal is to develop an application that is easy to use and provides effective error management.

Tools used

MS Word

Microsoft created Word, a word processing application. Initially available under the name Multi-Tool Word for Xenix systems, it was introduced on October 25, 1983. Later, versions were written for a number of other platforms, such as the following: web browsers (2010), iOS (2014), Android (2015), IBM PCs running DOS (1983), Apple Macintosh running the Classic Mac OS (1985), AT&T UNIX PC (1985), Atari ST (1988), OS/2 (1989), Microsoft Windows (1989), SCO Unix (1990), macOS (2001), and Atari ST (1988).

Word's commercial editions can be purchased with a perpetual license or as part of a Microsoft 365 subscription. Word is licensed as both a stand-alone application and a part of the Microsoft 365 suite of software *(Microsoft word 2024)*.

Figure 1:MS Word

IDLE

One tool for Python programming is the lightweight, user-friendly Python IDLE (Integrated Development and Learning Environment). The standard Python implementation has incorporated IDLE, an integrated development environment, since version 1.5.2b1. It is an optional component of the Python

packaging seen in many Linux distributions. Python and the Tkinter GUI toolkit are utilized throughout (*Idle software in Python - Javatpoint*).

Figure 2: Idle

Draw.io

Draw.io is a proprietary tool for creating charts and diagrams. You can design a custom layout or use the software's automatic layout tool. They offer hundreds of visual elements and a wide variety of shapes to create a unique diagram or chart. The drag-and-drop functionality facilitates the creation of visually appealing charts and diagrams.

Depending on your needs, Draw.io offers choices for saving saved charts on a server, in the cloud, or on network storage at a data centre. Their monthly fees range from $5.00 for ten users to $577.50 for five thousand members (*What is Draw.io? 2024*).

Figure3: Draw.io

Algorithm

An algorithm is a process for carrying out a calculation or problem-solving. In hardware- or software-based routines, algorithms function as a precise set of instructions that carry out predetermined operations one after the other.

All branches of information technology heavily rely on algorithms. An algorithm is typically used in mathematics, computer science, and computer programming to describe a brief process that resolves a recurring issue. Algorithms are essential to automated systems because they provide as guidelines for processing data (Gillis, *What is an algorithm?: TechTarget* 2024).

The Algorithm for my program is given below:

Step 1: Start

- Begin the program.

Step 2: Display Welcome Message

- Print a welcome message to the user.

- Store the user's name (optional, if needed for invoices).

Step 3: Display Options

- Present the following options to the user:
    1. Show all available furniture
    2. Purchase furniture from manufacturer
    3. Sell furniture to customer
    4. Exit

Step 4: Get User Choice

- Prompt the user to enter their choice (1-4).

Step 5: Handle User Choice

- If choice is 1 (Show all available furniture):
    - Call display_inventory() to display the current inventory.
    - Repeat from Step 3 (go back to displaying options).
- If choice is 2 (Purchase furniture):
    - Display message "Purchase furniture".
    - Prompt for the employee's name.
    - Prompt for the ID of the furniture to buy.
    - Prompt for the quantity of the product the employee wants.
    - Search for the furniture in the inventory:
        - If found:
            - Update the inventory quantity.
            - Print the purchase bill using generate_bill().
        - If not found:
            - Print "Furniture ID not found in inventory."
    - Repeat from Step 3.

- If choice is 3 (Sell furniture):
    - Display message "Selling furniture to customers".
    - Prompt for the customer's name.
    - Initialize an empty list for items to buy.
    - While True loop to allow multiple purchases:
        - Prompt for the furniture ID to purchase.
        - Prompt for the quantity to buy.
        - Search for the furniture in the inventory:
            - If found:
                - Check if the quantity is available.
                - If sufficient, update inventory and add item to the list.
                - Ask if the customer wants to buy more items.
                - If yes, repeat from this point.
                - If no, proceed to print the sales bill.
            - If not found, print "Furniture ID not found in inventory."
    - Print the sales bill using generate_bill().
    - Repeat from Step 3.
- If choice is 4 (Exit):
    - Print "Thank you for using the Furniture Inventory System. Goodbye!".
    - End the program.

Step 6: End

- The program concludes.

Flowchart

A flowchart is a visual diagram used to represent processes, systems, or algorithms. Commonly employed across various fields, it helps document, analyse, plan, and communicate complex processes clearly. Flowcharts use shapes like rectangles, ovals, and diamonds to depict steps, and arrows to indicate flow and sequence (Charntaweekhun & Wangsiripitak, *Visual programming using flowchart* 2006).

Figure 4: Flowchart

Pseudocode

Pseudocode is a step-by-step description of an algorithm written in plain English, not in a programming language. It serves as an intermediate step between the idea and its implementation in code, aimed at human understanding rather than machine interpretation (Oda et al., 2015).

Pseudocode for main.py

```
BEGIN MAIN
    WHILE TRUE
        DISPLAY "Welcome to the Furniture Inventory System!"
        DISPLAY "1. Show Inventory"
        DISPLAY "2. Purchase Furniture"
        DISPLAY "3. Sell Furniture"
        DISPLAY "4. Exit"

        ASSIGN choice GET INPUT "Enter your choice (1-4): "

        IF choice EQUALS '1' THEN
            CALL display_inventory
        END IF

        IF choice EQUALS '2' THEN
            CALL buy_furniture
        END IF
```

```
        IF choice EQUALS '3' THEN
            CALL sell_furniture
        END IF

        IF choice EQUALS '4' THEN
            DISPLAY "Thank you for using the Furniture Inventory System. Goodbye!"
            BREAK
        END IF

        DISPLAY "Invalid choice. Please enter a number between 1 and 4."
    END WHILE
END MAIN

IF _name_ EQUALS "_main_" THEN
    CALL main
END IF
```

Pseudocode for read.py

```
BEGIN load_inventory
    ASSIGN inventory TO NEW LIST

    TRY
```

```
        ASSIGN file GET OPEN 'inventory.txt' IN READ MODE

        FOR EACH line IN file DO
            ASSIGN parts GET SPLIT TRIM(line) BY ', '
            ASSIGN item TO NEW DICTIONARY
            ASSIGN item["id"] TO parts[0]
            ASSIGN item["manufacturer"] TO parts[1]
            ASSIGN item["name"] TO parts[2]
            ASSIGN item["quantity"] TO CONVERT TO INT(parts[3])
            ASSIGN item["price"] TO CONVERT TO
FLOAT(REPLACE(parts[4], '$', ''))
            CALL inventory.APPEND(item)
        END FOR

    EXCEPT FileNotFoundError
        DISPLAY "Inventory file not found. Starting with an empty inventory."
    END TRY

    RETURN inventory
END load_inventory
```

Pseudocode for write.py

```
ASSIGN VAT_RATE TO 0.13
```

```
ASSIGN SHIPPING_COST TO 50


BEGIN save_inventory
    ASSIGN file GET OPEN 'inventory.txt' IN WRITE MODE


    FOR EACH item IN inventory DO
        ASSIGN formatted_string TO CONCAT(item['id'], ', ', item['manufacturer'], ', ', item['name'], ', ', item['quantity'], ', $', item['price'], '\n')
        CALL file.WRITE(formatted_string)
    END FOR


END save_inventory


BEGIN create_invoice
    ASSIGN file GET OPEN filename IN WRITE MODE


    CALL file.WRITE("BRJ Furniture Stores - Invoice\n")
    CALL file.WRITE(CONCAT("Date: ", invoice_data['date'], '\n'))
    CALL file.WRITE(CONCAT("Name: ", invoice_data['name'], '\n\n'))


    CALL file.WRITE("+----+-----------------------+----------------+----------+-------+------------+\n")
    CALL file.WRITE("| ID | Manufacturer          | Product Name   | Quantity | Price | Total      |\n")
```

```
    CALL file.WRITE("+----+----------------------+----------------+----------+-------
-+------------+\n")


    ASSIGN subtotal TO 0


    FOR EACH item IN invoice_data['items'] DO
        ASSIGN total TO item['quantity'] * item['price']
        CALL file.WRITE(CONCAT("| ", item['id'], " | ",
item['manufacturer'][:23], " | ", item['name'][:15], " | ", item['quantity'], " | $",
item['price'], " | $", total, " |\n"))
        subtotal ASSIGN subtotal + total
    END FOR


    CALL file.WRITE("+----+----------------------+----------------+----------+-------
-+------------+\n")


    ASSIGN vat_amount TO subtotal * VAT_RATE
    ASSIGN total_with_vat TO subtotal + vat_amount
    ASSIGN total_amount TO total_with_vat + SHIPPING_COST


    CALL file.WRITE(CONCAT("\nSubtotal: $", subtotal, "\n"))
    CALL file.WRITE(CONCAT("VAT (", INT(VAT_RATE * 100), "%): $",
vat_amount, "\n"))
    CALL file.WRITE(CONCAT("Total with VAT: $", total_with_vat, "\n"))
    CALL file.WRITE(CONCAT("Shipping Cost: $", SHIPPING_COST, "\n"))
    CALL file.WRITE(CONCAT("Total Amount: $", total_amount, "\n"))
```

```
    CALL file.WRITE("\nThank you for your business!\n")


END create_invoice


Pseudocode for operation.py

BEGIN buy_furniture
    ASSIGN name GET INPUT "Enter the employee name: "
    ASSIGN item_id GET INPUT "Enter the ID of the furniture to purchase: "
    ASSIGN quantity GET CONVERT TO INT(INPUT "Enter the quantity to
purchase: ")


    FOR EACH item IN furniture_data DO
        IF item["id"] EQUALS item_id THEN
            item["quantity"] ASSIGN item["quantity"] + quantity
            BREAK
        END IF
    END FOR
    ELSE
        DISPLAY "Furniture ID not found in inventory."
        RETURN
    END ELSE


    ASSIGN bill_details TO NEW DICTIONARY
    ASSIGN bill_details["Date"] TO GET CURRENT DATE AND TIME AS
STRING
```

```
    ASSIGN bill_details["Name"] TO name
    ASSIGN bill_details["Items"] TO NEW LIST
    ASSIGN item_details TO NEW DICTIONARY
    ASSIGN item_details["id"] TO item["id"]
    ASSIGN item_details["manufacturer"] TO item["manufacturer"]
    ASSIGN item_details["product name"] TO item["product name"]
    ASSIGN item_details["quantity"] TO quantity
    ASSIGN item_details["price"] TO item["price"]
    CALL bill_details["Items"].APPEND(item_details)

    ASSIGN filename TO CONCAT("buy_bill_", item_id, "_", GET
CURRENT DATE AND TIME AS STRING, ".txt")
    CALL generate_bill(bill_details, filename, is_purchase=True)

    DISPLAY "Purchase completed and inventory updated."
    CALL save_furniture_data(furniture_data, file_path)
END buy_furniture

BEGIN sell_furniture
    ASSIGN name GET INPUT "Enter the customer name: "
    ASSIGN items_to_buy TO NEW LIST

    WHILE TRUE DO
        ASSIGN item_id GET INPUT "Enter the ID of the furniture to
purchase: "
```

```
    ASSIGN quantity GET CONVERT TO INT(INPUT "Enter the quantity to purchase: ")


    FOR EACH item IN furniture_data DO
        IF item["id"] EQUALS item_id THEN
            IF quantity GREATER THAN item["quantity"] THEN
                DISPLAY CONCAT("Only ", item["quantity"], " pieces available in stock.")
                RETURN
            END IF
            item["quantity"] ASSIGN item["quantity"] - quantity
            ASSIGN item_details TO NEW DICTIONARY
            ASSIGN item_details["id"] TO item["id"]
            ASSIGN item_details["manufacturer"] TO item["manufacturer"]
            ASSIGN item_details["product name"] TO item["product name"]
            ASSIGN item_details["quantity"] TO quantity
            ASSIGN item_details["price"] TO item["price"]
            CALL items_to_buy.APPEND(item_details)
            BREAK
        END IF
    END FOR
    ELSE
        DISPLAY "Furniture ID not found in inventory."
        RETURN
    END ELSE
```

```
        ASSIGN another_item GET INPUT "Do you want to buy another item? (y/n): "
        IF another_item NOT EQUALS 'y' THEN
            BREAK
        END IF
    END WHILE

    ASSIGN invoice_details TO NEW DICTIONARY
    ASSIGN invoice_details["Date"] TO GET CURRENT DATE AND TIME AS STRING
    ASSIGN invoice_details["Name"] TO name
    ASSIGN invoice_details["Items"] TO items_to_buy

    ASSIGN filename TO CONCAT("sales_invoice_", name, "_", GET CURRENT DATE AND TIME AS STRING, ".txt")
    CALL generate_bill(invoice_details, filename, is_purchase=False)

    DISPLAY "Sale completed and inventory updated."
    CALL save_furniture_data(furniture_data, file_path)
END sell_furniture
```

Data Structure

Data structures organize data for efficient access and are essential in programming. Python simplifies learning these fundamentals compared to other languages.

List

Data structures organize data for efficient access and are essential in programming. Python simplifies learning these fundamentals compared to other languages.

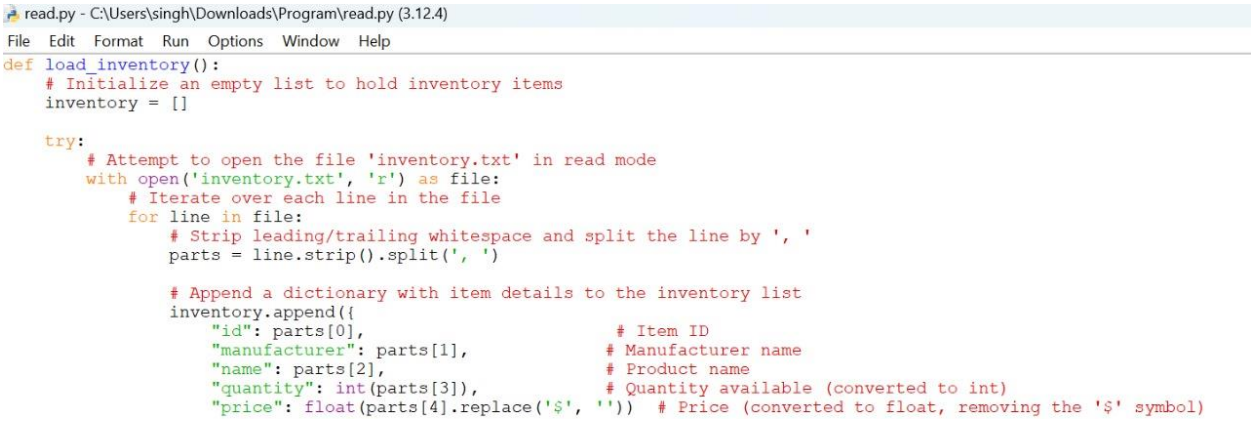

Figure 5: Inventory list

It is created within the load_inventory function (in Read. py) and holds all the items of the inventory.

Dictionaries

Dictionaries are a powerful data structure in Python, designed for storing key-value pairs. They mimic real-world data arrangements where each key is associated with a specific value, making them ideal for efficient data lookup and retrieval.

```
# Generate and save the purchase invoice
bill_details = {
    "Date": datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
    "Name": name,
    "Items": [
        {
            "id": item["id"],
            "manufacturer": item["manufacturer"],
            "product name": item["product name"],
            "quantity": quantity,
            "price": item["price"]
        }
```

Figure 6: Dictionary of Invoice Details

This is a dictionary containing description of an invoice such as date, name of the customer, or type of items bought.

Strings in Python

A string in Python is a data structure that represents a sequence of characters. It is immutable, meaning once created, it cannot be changed. Strings are commonly used for handling text data like names, addresses, and other text-based information in various applications.

```
def load_inventory():
    # Initialize an empty list to hold inventory items
    inventory = []

    try:
        # Attempt to open the file 'inventory.txt' in read mode
        with open('inventory.txt', 'r') as file:
            # Iterate over each line in the file
            for line in file:
                # Strip leading/trailing whitespace and split the line by ', '
                parts = line.strip().split(', ')
```

Figure 7: File handling

Filename is a string which is why info about the file is stored in same string variable for purpose of reading or writing the file.

Integer/Float

```
"quantity": int(parts[3]),            # Quantity available (converted to int)
"price": float(parts[4].replace('$', ''))  # Price (converted to float, removing the '$' symbol)
```

Figure 8: integers and float

Prices are kept as floating-point numbers, and item

quantities are stored as integers.

Program

The purpose of this program is to improve the process of buying and selling furniture at BRJ Furniture Store.

Implementation of the program

After running the program, a welcome message, store location, and phone number are displayed, followed by four options to choose from. The first option lists all available furniture in the store. The second option shows furniture purchased from manufacturers. The third option facilitates selling furniture to customers. Finally, the program provides an exit option.

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: C:\Users\singh\Downloads\Program\main.py

Welcome to the Furniture Inventory System!
1. Show Inventory
2. Purchase Furniture
3. Sell Furniture
4. Exit
Enter your choice (1-4): 
```

Figure 9: Implementation

```
. Exit
nter your choice (1-4): 1
----+-------------------------+-----------------+----------+--------+
 ID | Manufacturer            | Product Name    | Quantity | Price  |
----+-------------------------+-----------------+----------+--------+
 F001 | Wooden Chair           | Seating         |      134 | $1500.00 |
 F002 | Glass Table            | Table           |        5 | $4500.00 |
 F003 | Leather Sofa           | Seating         |        3 | $8500.00 |
 F004 | Metal Cabinet          | Storage         |        7 | $3000.00 |
 F005 | Queen Bed              | Bedroom         |        7 | $9500.00 |
 F006 | Dining Table           | Table           |       16 | $6000.00 |
----+-------------------------+-----------------+----------+--------+

elcome to the Furniture Inventory System!
. Show Inventory
```

Figure 10: Available Furniture

Show the Purchase and Sale of Furniture

Purchasing Furniture from Manufacturer

To buy furniture, select option 2. During the purchase process, you will be prompted to provide the name and ID of the furniture, along with the quantity to purchase. This allows the program to update the inventory accordingly.

```
Welcome to the Furniture Inventory System!
1. Show Inventory
2. Purchase Furniture
3. Sell Furniture
4. Exit
Enter your choice (1-4): 2
Enter your name: Tapendra
Enter the ID of the furniture to purchase: F002
Enter the quantity to purchase: 2
2 units of Table added to the inventory.
```

Figure 11: Purchasing Furniture

Selling Furniture to Customers

Once the furniture has been purchased from the manufacturer, it can be sold to customers. To initiate a sale, select option 3. You will need to provide valid information for the transaction. Customers can also purchase multiple pieces of furniture at once, allowing for a more flexible shopping experience.

```
Welcome to the Furniture Inventory System:
1. Show Inventory
2. Purchase Furniture
3. Sell Furniture
4. Exit
Enter your choice (1-4): 3
Enter customer's name: Tsering Tamang
Enter the ID of the furniture to sell: F001
Enter the quantity to sell: 4
Sell another item? (y/n): y
Enter the ID of the furniture to sell: F004
Enter the quantity to sell: 3
Sell another item? (y/n): n
```

Figure 12: selling furniture to the customer

Creation of a TXT File

To manage the inventory, a text file named **Furniture_inventory.txt** will be created. This file will store the details of the furniture items available in the store, allowing for easy access and updates to the inventory data.

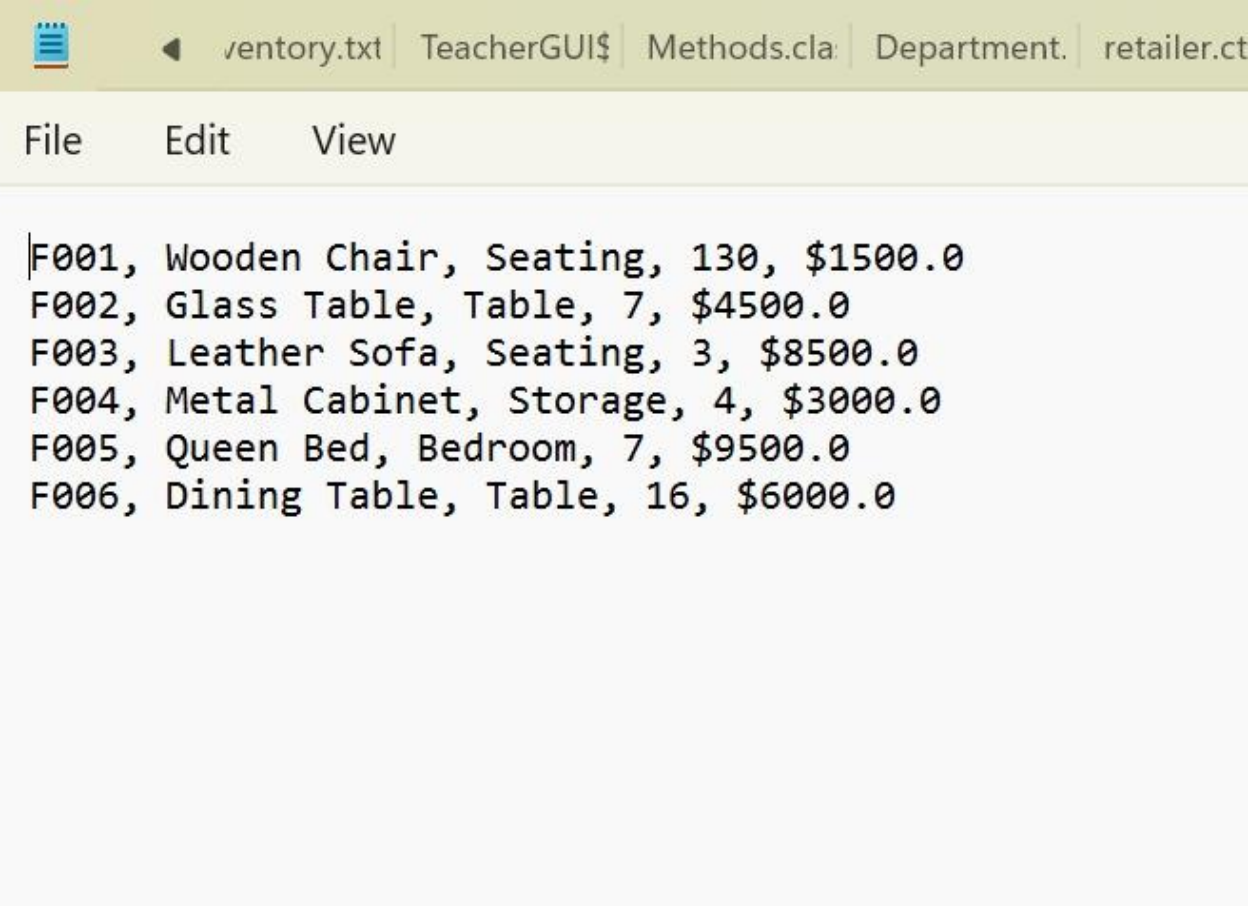

Figure 13: Txt file

Opening text and show the bill

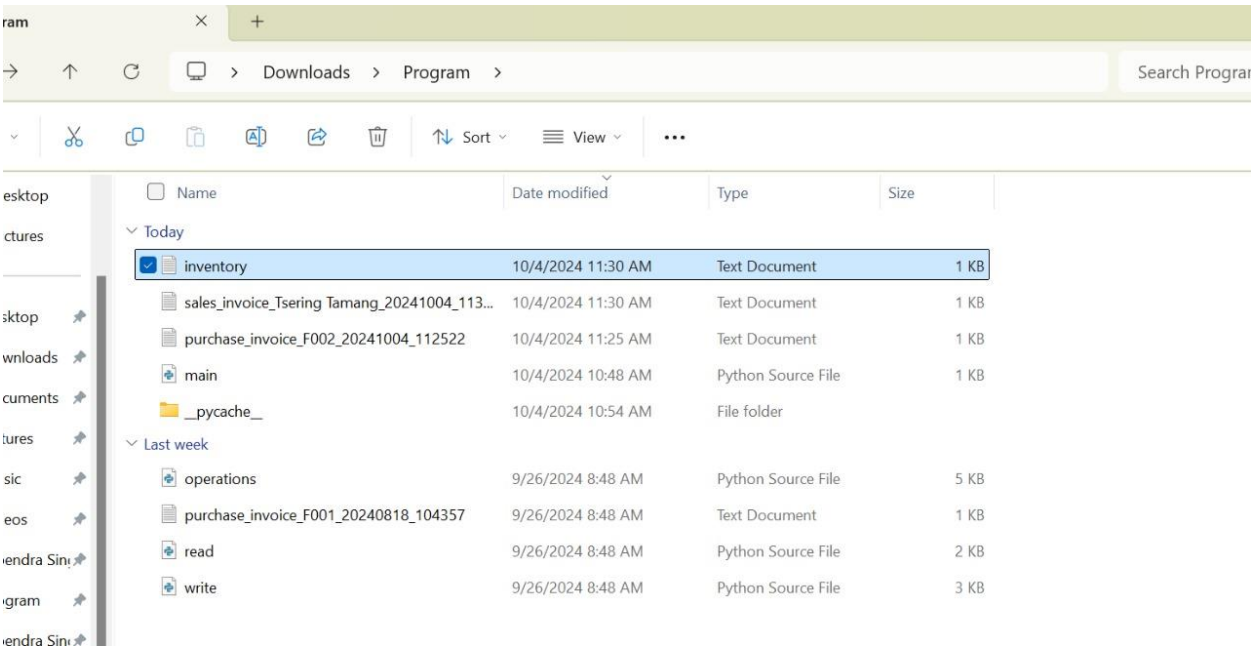

Figure 14: opening Text fie

```
BRJ Furniture Stores - Invoice
Date: 2024-10-04 11:25:22.801636
Name: Tapendra

+----+-------------------------+-----------------+----------+--------+------------+
| ID | Manufacturer            | Product Name    | Quantity | Price  | Total      |
+----+-------------------------+-----------------+----------+--------+------------+
| F002 | Glass Table             | Table           |        2 | $4500.00 | $   9000.00 |
+----+-------------------------+-----------------+----------+--------+------------+

Subtotal: $9000.00
VAT (13%): $1170.00
Total with VAT: $10170.00
Shipping Cost: $50.00
Total Amount: $10220.00

Thank you for your business!
```

Figure 15: purchase invoice

```
BRJ Furniture Stores - Invoice
Date: 2024-10-04 11:30:28.212746
Name: Tsering Tamang

+----+-------------------------+----------------+----------+--------+------------+
| ID | Manufacturer            | Product Name    | Quantity | Price  | Total      |
+----+-------------------------+----------------+----------+--------+------------+
| F001 | Wooden Chair            | Seating         |        4 | $1500.00 | $   6000.00 |
| F004 | Metal Cabinet           | Storage         |        3 | $3000.00 | $   9000.00 |
+----+-------------------------+----------------+----------+--------+------------+

Subtotal: $15000.00
VAT (13%): $1950.00
Total with VAT: $16950.00
Shipping Cost: $50.00
Total Amount: $17000.00

Thank you for your business!
```

Figure 16: sales invoice

Termination of the program after selecting an option

```
1. Show Inventory
2. Purchase Furniture
3. Sell Furniture
4. Exit
Enter your choice (1-4): 4
Thank you for using the Furniture Inventory System. Goodbye!
```

Testing

Test 1:

Show implementation of try, except

Table 1: implementation

Objective	To show the implementation of try and except

Action	Run the program in IDLE and enter an invalid number
Expected result	Invalid option would show up as the message.
Actual	Invalid option was shown
Conclusion	the test was successful

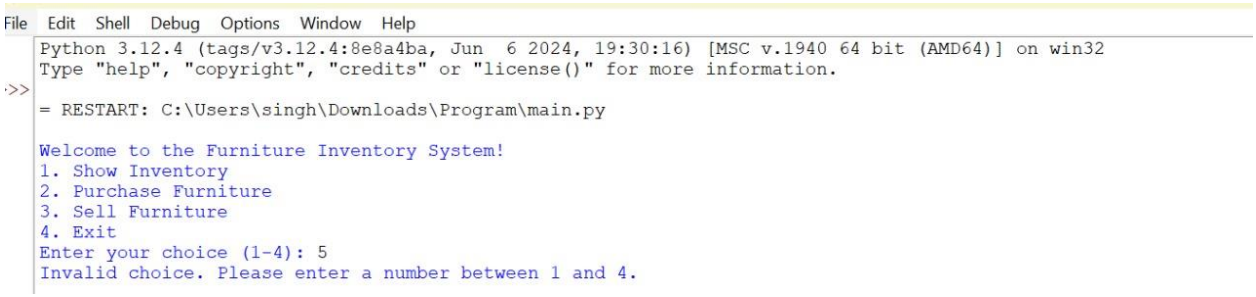

Figure 17: Invalid option

Test 2:

Selection purchase and sell of furniture.

Table 2: purchase and sell of furniture

Objective	Selection purchase and selling of furniture
Action	Trying to buy furniture by providing non-existing input and selling furniture by providing negative input
Expected result	The error output should be shown
Actual result	Error output is shown.
Conclusion	The test is successful

```
Welcome to the Furniture Inventory System!
1. Show Inventory
2. Purchase Furniture
3. Sell Furniture
4. Exit
Enter your choice (1-4): 2
Enter your name: tapendra
Enter the ID of the furniture to purchase: 3
Enter the quantity to purchase: 4
Furniture with ID 3 not found.
```

Figure 18: Entering non-existed input

```
========== RESTART: C

Welcome to the Furniture Inventory System!
1. Show Inventory
2. Purchase Furniture
3. Sell Furniture
4. Exit
Enter your choice (1-4): -4
Invalid choice. Please enter a number between 1 and 4.
```

Figure 19: Entering negative input

Test 3:

File generation of purchasing process of furniture(s)

Table 3: invoice of purchasing

Objective	File generation of purchasing process of furniture
Action	Purchasing a product from manufacturer and showing the bill
Expected result	Bill should be generated

Actual result	Bill is generated
Conclusion	The test is successful

```
Welcome to the Furniture Inventory System!
1. Show Inventory
2. Purchase Furniture
3. Sell Furniture
4. Exit
Enter your choice (1-4): 2
Enter your name: tapendra
Enter the ID of the furniture to purchase: F003
Enter the quantity to purchase: 4
4 units of Seating added to the inventory.
```

Figure 20: product purchasing

```
BRJ Furniture Stores - Invoice
Date: 2024-10-04 11:42:28.384033
Name: tapendra

+----+-------------------------+-----------------+----------+--------+------------+
| ID | Manufacturer            | Product Name    | Quantity | Price  | Total      |
+----+-------------------------+-----------------+----------+--------+------------+
| F003 | Leather Sofa            | Seating         |        4 | $8500.00 | $  34000.00 |
+----+-------------------------+-----------------+----------+--------+------------+

Subtotal: $34000.00
VAT (13%): $4420.00
Total with VAT: $38420.00
Shipping Cost: $50.00
Total Amount: $38470.00

Thank you for your business!
```

Figure 21: invoice for purchasing

Test 4:

File generation of selling process of furniture(s) (selling multiple furniture)

Table 4: invoice of selling

Objective	File generation of selling process of furniture
Action	Selling of furniture to a customer and generate bill
Expected result	Bill should be generated
Actual Result	Bill is generated
Conclusion	The test is successful

```
Welcome to the Furniture Inventory System!
1. Show Inventory
2. Purchase Furniture
3. Sell Furniture
4. Exit
Enter your choice (1-4): 3
Enter customer's name: F001
Enter the ID of the furniture to sell: F001
Enter the quantity to sell: 4
Sell another item? (y/n): y
Enter the ID of the furniture to sell: F002
Enter the quantity to sell: 2
Sell another item? (y/n): n
```

Figure 22: product is sold

```
BRJ Furniture Stores - Invoice
Date: 2024-10-04 11:49:02.655240
Name: F001

+----+-------------------------+----------------+----------+--------+------------+
| ID | Manufacturer            | Product Name    | Quantity | Price  | Total      |
+----+-------------------------+----------------+----------+--------+------------+
| F001 | Wooden Chair            | Seating         |        4 | $1500.00 | $   6000.00 |
| F002 | Glass Table             | Table           |        2 | $4500.00 | $   9000.00 |
+----+-------------------------+----------------+----------+--------+------------+

Subtotal: $15000.00
VAT (13%): $1950.00
Total with VAT: $16950.00
Shipping Cost: $50.00
Total Amount: $17000.00

Thank you for your business!
```

Figure 23: invoice for selling product

Test 5:

Show the update in stock of furniture(s)

Table 5: update stock

Objective	Update the furniture
Action	After buying/selling furniture inventory
Expected result	Inventory should updated
Actual result	Inventory has not updated
Conclusion	The test was unsuccessful

```
Welcome to the Furniture Inventory System!
1. Show Inventory
2. Purchase Furniture
3. Sell Furniture
4. Exit
Enter your choice (1-4): 2
Enter your name: Rahul
Enter the ID of the furniture to purchase: F004
Enter the quantity to purchase: 4
4 units of Storage added to the inventory.
```

Figure 24: inventory has not updated

Conclusion

This has added a lot to my insight into how novice Python programmers make use of the language. I consider finishing the course a success because, with it, I went from knowing absolutely nothing about Python to developing an application for a furniture store. I struggled a lot to finish this project because it was the first project in the model we were given, and I didn't know too much about Python. My problems with variables, logic, and functions caused the application to fail even when the correct values were entered. I have also encountered a lot of software errors, syntax issues leading to data type loss, and logical flaws leading to the output of wrong results. Because of this problem, I sought my teacher's assistance.

Appendix

Code for main.py

```
from operations import display_inventory, buy_furniture, sell_furniture


def main():
    while True:
        # Display the menu options
```

```
    print("\nWelcome to the Furniture Inventory System!")
    print("1. Show Inventory")
    print("2. Purchase Furniture")
    print("3. Sell Furniture")
    print("4. Exit")

    # Get the user's choice
    choice = input("Enter your choice (1-4): ")

    # Handle the user's choice
    if choice == '1':
        display_inventory()
    elif choice == '2':
        buy_furniture()
    elif choice == '3':
        sell_furniture()
    elif choice == '4':
        print("Thank you for using the Furniture Inventory System. Goodbye!")
        break
    else:
        print("Invalid choice. Please enter a number between 1 and 4.")

# Entry point of the program
if _name_ == "_main_":
```

```
    main()
```

Code for read.py

```
def load_inventory():
    # Initialize an empty list to hold inventory items
    inventory = []

    try:
        # Attempt to open the file 'inventory.txt' in read mode
        with open('inventory.txt', 'r') as file:
            # Iterate over each line in the file
            for line in file:
                # Strip leading/trailing whitespace and split the line by ', '
                parts = line.strip().split(', ')

                # Append a dictionary with item details to the inventory list
                inventory.append({
                    "id": parts[0],                    # Item ID
                    "manufacturer": parts[1],          # Manufacturer name
                    "name": parts[2],                  # Product name
                    "quantity": int(parts[3]),         # Quantity available (converted
to int)
                    "price": float(parts[4].replace('$', ''))  # Price (converted to float,
removing the '$' symbol)
```

```
                })
    except FileNotFoundError:
        # If the file does not exist, print a message and continue with an empty inventory
        print("Inventory file not found. Starting with an empty inventory.")

    # Return the inventory list
    return inventory
```

Code for write.py

```
VAT_RATE = 0.13  # VAT rate of 13%
SHIPPING_COST = 50  # Fixed shipping cost

# Function to save the inventory data to a file
def save_inventory(inventory):
    # Open 'inventory.txt' in write mode
    with open('inventory.txt', 'w') as file:
        # Iterate over each item in the inventory
        for item in inventory:
            # Write each item's details to the file in a formatted string
            file.write(f"{item['id']}, {item['manufacturer']}, {item['name']}, {item['quantity']}, ${item['price']}\n")

# Function to create an invoice and save it to a file
def create_invoice(invoice_data, filename):
```

```
    # Open the specified file in write mode
    with open(filename, 'w') as file:
        # Write the header of the invoice
        file.write("BRJ Furniture Stores - Invoice\n")
        file.write(f"Date: {invoice_data['date']}\n")
        file.write(f"Name: {invoice_data['name']}\n\n")

        # Write the table headers
        file.write("+----+-----------------------+----------------+----------+--------+-----------+\n")
        file.write("| ID | Manufacturer          | Product Name    | Quantity | Price  | Total      |\n")
        file.write("+----+-----------------------+----------------+----------+--------+-----------+\n")

        subtotal = 0  # Initialize subtotal for invoice items

        # Iterate over each item in the invoice
        for item in invoice_data['items']:
            # Calculate the total cost for the current item
            total = item['quantity'] * item['price']
            # Write the item details and total cost to the file
            file.write(f"| {item['id']:2} | {item['manufacturer'][:23]:23} | {item['name'][:15]:15} | {item['quantity']:8} | ${item['price']:6.2f} | ${total:10.2f} |\n")
            # Add the item total to the subtotal
```

```
            subtotal += total

        # Write the table footer
        file.write("+----+------------------------+----------------+----------+--------+-----------+\n")

        # Calculate VAT and total amounts
        vat_amount = subtotal * VAT_RATE
        total_with_vat = subtotal + vat_amount
        total_amount = total_with_vat + SHIPPING_COST

        # Write the financial summary to the file
        file.write(f"\nSubtotal: ${subtotal:.2f}\n")
        file.write(f"VAT ({int(VAT_RATE * 100)}%): ${vat_amount:.2f}\n")
        file.write(f"Total with VAT: ${total_with_vat:.2f}\n")
        file.write(f"Shipping Cost: ${SHIPPING_COST:.2f}\n")
        file.write(f"Total Amount: ${total_amount:.2f}\n")
        file.write("\nThank you for your business!\n")
```

Code for operation.py

```
from Read import load_furniture_data
from write import save_furniture_data, generate_bill
import datetime
```

```
def buy_furniture(furniture_data, file_path):
    name = input("Enter the employee name: ")
    item_id = input("Enter the ID of the furniture to purchase: ")
    quantity = int(input("Enter the quantity to purchase: "))

    for item in furniture_data:
        if item["id"] == item_id:
            item["quantity"] += quantity
            break
    else:
        print("Furniture ID not found in inventory.")
        return

    # Generate and save the purchase invoice
    bill_details = {
        "Date": datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
        "Name": name,
        "Items": [
            {
                "id": item["id"],
                "manufacturer": item["manufacturer"],
                "product name": item["product name"],
                "quantity": quantity,
                "price": item["price"]
            }
```

```
        ]
    }
    filename =
f"buy_bill_{item_id}{datetime.datetime.now().strftime('%Y%m%d%H%M%S'
)}.txt"
    generate_bill(bill_details, filename, is_purchase=True)

    print("Purchase completed and inventory updated.")
    save_furniture_data(furniture_data, file_path)

def sell_furniture(furniture_data, file_path):
    name = input("Enter the customer name: ")
    items_to_buy = []
    while True:
        item_id = input("Enter the ID of the furniture to purchase: ")
        quantity = int(input("Enter the quantity to purchase: "))

        for item in furniture_data:
            if item["id"] == item_id:
                if quantity > item["quantity"]:
                    print(f"Only {item['quantity']} pieces available in stock.")
                    return
                item["quantity"] -= quantity
                items_to_buy.append({
                    "id": item["id"],
```

```
                "manufacturer": item["manufacturer"],
                "product name": item["product name"],
                "quantity": quantity,
                "price": item["price"]
            })
            break
    else:
        print("Furniture ID not found in inventory.")
        return

    another_item = input("Do you want to buy another item? (y/n): ").lower()
    if another_item != 'y':
        break

# Generate and save the sales invoice
invoice_details = {
    "Date": datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
    "Name": name,
    "Items": items_to_buy
}
filename = f"sales_invoice_{name}{datetime.datetime.now().strftime('%Y%m%d%H%M%S')}.txt"
generate_bill(invoice_details, filename, is_purchase=False)
```

```
    print("Sale completed and inventory updated.")
    save_furniture_data(furniture_data, file_path)
```

References

Charntaweekhun, K. and Wangsiripitak, S. (2006) ‘Visual programming using flowchart’, *2006 International Symposium on Communications and Information Technologies*, pp. 1062–1065. doi:10.1109/iscit.2006.339940.

Gillis, A.S. (2024) *What is an algorithm?: TechTarget*, *WhatIs*. Available at: https://www.techtarget.com/whatis/definition/algorithm (Accessed: 12 October 2024).

Idle software in Python - Javatpoint (no date) *www.javatpoint.com*. Available at: https://www.javatpoint.com/idle-software-in-python (Accessed: 12 October 2024).

Microsoft word (2024) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Microsoft_Word (Accessed: 12 October 2024).

What is Draw.io? (2024) *Computer Hope*. Available at: https://www.computerhope.com/jargon/d/drawio.htm (Accessed: 12 October 2024).

Oda, Y. *et al.* (2015) ‘Learning to generate Pseudo-Code from source code using Statistical Machine Translation’, *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)* [Preprint]. doi:10.1109/ase.2015.36.