A logo with blue and grey text

Description automatically generated

**Module Code & Module Title:**

**CS4051NT Fundamentals of Computing**

**Assessment Weightage & Type:**

**60% Individual Coursework**

**Year and Semester:**

**2024 Spring**

**Student Name: Abhishek Sharma**

**London Met ID: 23056498**

**College ID: NP05CP4S240023**

**Assignment Due Date: August 18, 2024**

**Word Count:**

*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 mark of zero will be awarded.*

Table of Contents

[1 Introduction 6](#_Toc173998404)

[1.1 Introduction to Python 6](#_Toc173998405)

[1.2 Features of Python 7](#_Toc173998406)

[1.3 Introduction to IDLE 8](#_Toc173998407)

[1.4 Introduction to coursework 9](#_Toc173998408)

[1.5 Goals and Objectives 9](#_Toc173998409)

[2 Algorithm 10](#_Toc173998410)

[2.1 Algorithm of overall code 11](#_Toc173998411)

[2.2 Detailed Explanation of each Functions 13](#_Toc173998412)

[2.2.1 main() function 13](#_Toc173998413)

[2.2.2 read\_furniture\_data() function 13](#_Toc173998414)

[2.2.3 read\_furniture\_products() function 13](#_Toc173998415)

[2.2.4 read\_inventory() function 14](#_Toc173998416)

[2.2.5 write\_inventory(inventory) function 14](#_Toc173998417)

[2.2.6 generate\_invoice(transactions) function 14](#_Toc173998418)

[2.2.7 add\_new\_product(furniture\_id, manufacturer, product\_name, quantity, price) function 15](#_Toc173998419)

[2.2.8 update\_inventory(furniture\_id, quantity\_change) function 15](#_Toc173998420)

[2.2.9 order\_furniture(furniture\_id, quantity, employee\_name) function 15](#_Toc173998421)

[2.2.10 sell\_furniture(customer\_name, furniture\_id, quantity) function 16](#_Toc173998422)

[3 Pseudocode 17](#_Toc173998423)

[3.1 Pseudocode for main.py code 17](#_Toc173998424)

[3.2 Pseudocode for read.py 20](#_Toc173998425)

[3.3 Pseudocode for write.py 22](#_Toc173998426)

[3.4 Pseudocode for operation.py 23](#_Toc173998427)

[4 Flowchart 26](#_Toc173998428)

[5 Data Structures 27](#_Toc173998429)

[6 Program 28](#_Toc173998430)

[7 Testing 28](#_Toc173998431)

[8 Conclusion 29](#_Toc173998432)

[9 References 30](#_Toc173998433)

[10 Appendix 31](#_Toc173998434)

**Table of Figures:**

[Figure 1: Python Logo 6](#_Toc173080570)

[Figure 2: IDLE Logo 8](#_Toc173080571)

[Figure 3: Pictorial example of Algorithm 10](#_Toc173080572)

**Table of Tables:**

# Introduction

The "Fundamentals of Computing" module (CS4051NT), led by Mr. Ajayraj Bhattrai, focuses on Python programming, and covers essential computing concepts including, algorithms, flowcharts, and software development practices. This semester-long course assesses our performance through coursework, which accounts for 60% of our grade, and an in-course test, making up the remaining 40%. Designed to build our foundational knowledge and enhance critical thinking skills, the module combines lectures, tutorials, workshops, practical assignments, and coursework to provide a comprehensive learning experience in computing.

## Introduction to Python

Python is an admired general purpose; high level language with multifunctional uses and is acknowledged for its ease of use. Standardized by Guido van Rossum in 1991 and consistently developed by the Python Software Foundation, Python was designed and fine-tuned to be as readable and understandable, with simple and elegant syntactic structure, which would let the programmers express any idea in a clean, concise manner. Another significant aspect of Python is that it enables the quick development cycle where the members deliver fast, and it is amenable to integration. Over time, Python has evolved into two major versions: Two main groups of the programming language: Python 2 and Python 3 which differ in the features they provide, the capabilities they offer and the paradigms they support. However, both versions are still in use in multiple spheres ranging from web development, scientific computing, AI, and automation. (Geeks For Geeks, 2024)

A blue and yellow snake logo

Description automatically generated

Figure 1: Python Logo

## Features of Python

**Free and Open Source**

Python is an open-source language that can be downloaded from the official Python website and can be downloaded by just clicking on the Download Python link. Since it is an open source, the code is easily accessible to everyone for making modifications. It is free from legal protection and can be downloaded, used, and shared with anyone and everyone.

**Easy to Code**

Python is a great programming language that is relatively easier to learn than languages such as C, C#, JavaScript or Java. It will take you a few hours or days to learn how to program in Python if you are a beginner. It is intended for new users as well as advanced developers in the field and network engineering.

**Easy to Read**

Python is easy to learn since its structures are quite basic and uncluttered. Thus, the code structure in Python is very clear, and instead of semicolons or brackets, the code blocks are identified by indentations.

**Object-Oriented Language**

Python also supports Object-Oriented Programming which comprises of the notions of classes and object encapsulation. Due to this, it is an ideal option for creating intricate applications and programs.

**GUI Programming Support**

There are GUI modules available for Python like PyQt5 and PyQt4, wxPython, Tkinter and so on. In the present scenario, GUI frameworks are widely used, and out of all, PyQt5 is in great demand for developing graphical applications. And many more. (Geeks for geeks, 2024)

## Introduction to IDLE

IDLE, Interactive Development and Learning Environment for Python runs on Windows Unix as well as MacOS. It has a feature of a Python shell for entering code and the prompt for input, output, and exceptional conditions are in the colour. There are nice additional options that you can see in the text editor such as undo, highlighter for Python code and tips when writing functions. You can find specific text to replace it with or in files with a particular name. Like many programming languages, IDLE also provides the debugger to track down the errors in your code and allows you to step through your code to see how the program runs. that is convenient to work with adding that it has settings and tools included within. (Python Software Foundation, 2024)

A close-up of a paper

Description automatically generated

Figure 2: IDLE Logo

## Introduction to coursework

This coursework for the "Fundamentals of Computing" module (CS4051NT) focuses on developing a Python program to manage inventory and transactions for a furniture store. The program will read, and update information stored in a text file, displaying available furniture, and generating transaction notes for both purchases from manufacturers and sales to customers. It will incorporate features like error handling, input validation, and the calculation of VAT for sales transactions. The coursework emphasizes modular programming, using appropriate data structures for efficient data management, and includes documentation, algorithm development, and testing to ensure functionality and reliability.

## Goals and Objectives

The objectives of our coursework for the "Fundamentals of Computing" module are designed to achieve comprehensive understanding and practical skills in Python programming and software development. Our goals include:

* **Develop a Functional Inventory Management System:** Create a Python program to manage furniture inventory for a store, allowing updates for sales and orders directly impacting stock levels.
* **Implement Modular Programming:** Structure the program into separate functions for managing inventory, processing transactions, and generating invoices to ensure code reusability and clarity.
* **Utilize Effective Data Structures:** Employ appropriate data structures like dictionaries and lists in Python to efficiently store and manipulate furniture data, ensuring optimal performance for inventory operations.
* **Document and Present:** Provide clear documentation including algorithm descriptions, pseudocode, and flowcharts to illustrate program logic and functionality. Include a testing section to validate program behavior under different scenarios.

These objectives focus on developing practical Python programming skills while addressing the specific requirements of managing furniture inventory and transaction processing as outlined in our coursework.

# Algorithm

An algorithm is a set of directions used to accomplish a certain objective or solve a problem. While algorithms are mostly performed by computers, people also employ them in their daily lives. In real life, we are following an algorithm each time we carry out a certain consecutive operation, from tying the shoelaces to making coffee in the morning, for instance.

In other word it simply refers to a step-by-step procedure that can be used to solve a problem in computer science. Algorithms are used in search engines, music streaming, navigation systems, and many other systems and applications that are present in computers. (Nikolopoulou, 2024)

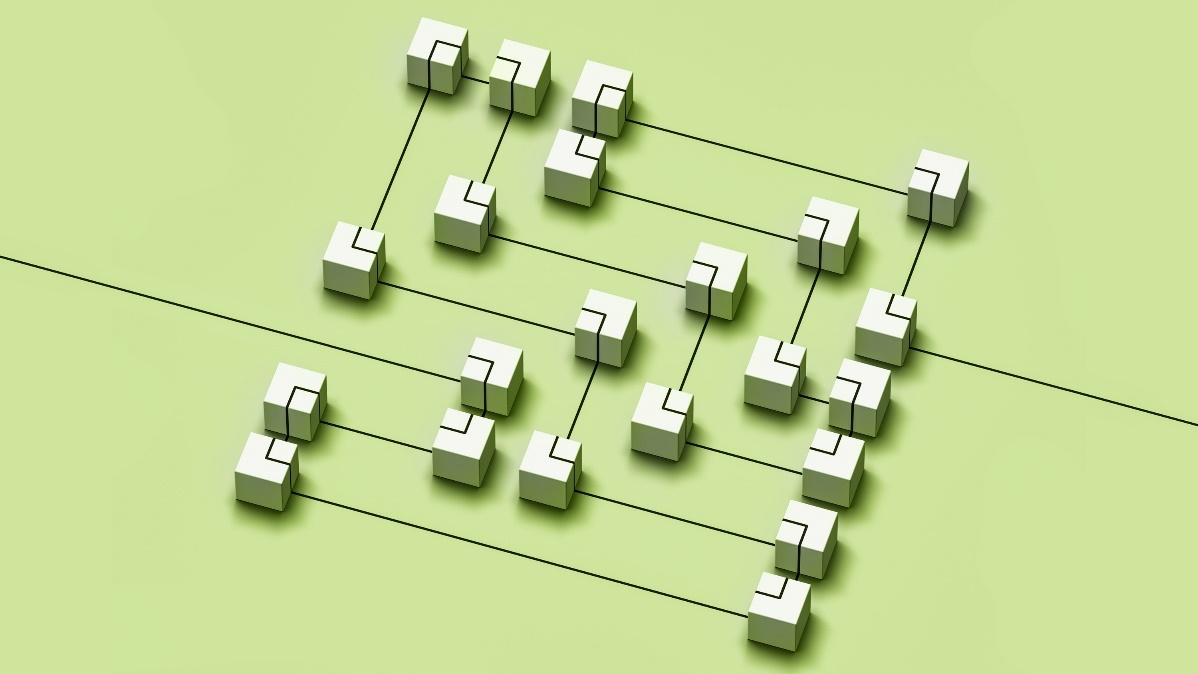


Figure 3: Pictorial example of Algorithm

## Algorithm of overall code

Step 1: Start

Step 2: Display the main menu with the following options:

Display available furniture

Purchase furniture from manufacturer

Sell furniture to customer

Exit

Step 3: Prompt the user for a choice, choice:

Step 4: Handle the user's choice:

if Choice is 1: Display available furniture

Call read\_furniture\_data() to read and display the furniture data from inventory.txt.

elif Choice is 2: Purchase furniture from manufacturer

Enter a loop for ordering furniture:

Display the current inventory by calling read\_furniture\_products().

Prompt the user to enter the furniture ID and quantity to order.

Prompt the user to enter the employee's name.

Call order\_furniture(furniture\_id, quantity, employee\_name) to handle the order.

Ask if the user wants to enter another transaction. If the user inputs "no", break the loop.

Generate an invoice for the transactions by calling generate\_invoice(transactions).

elif Choice is 3: Sell furniture to customer

Enter a loop for selling furniture:

Prompt the user to enter the customer's name.

Display the current inventory by calling read\_inventory().

Display the current inventory by calling read\_furniture\_products().

Prompt the user to enter the furniture ID and quantity to sell.

Call sell\_furniture(customer\_name, furniture\_id, quantity) to handle the sale.

Ask if the user wants to sell another product. If the user inputs "no", break the loop.

elif Choice is 4: Exit

Print a thank you message and exit the program.

else Invalid Choice:

Print an error message and prompt the user to try again.

Step 5: End

## Detailed Explanation of each Functions

### main() function

The main execution of the program is encapsulated in the main() function. This function provides a menu-driven interface for users to interact with the BRJ Furniture Store Management System. The available operations are displaying available furniture, purchasing furniture from the manufacturer, selling furniture to a customer, and exiting the program. The function handles user input, manages transactions, and generates invoices as needed. The program continues to prompt the user for choices until the user decides to exit by choosing option 4. When the script is run directly, the main() function is called to start the program.

### read\_furniture\_data() function

The purpose of the read\_furniture\_data function is to read and display all the furniture data from the file inventory.txt. It starts by attempting to open the file in read mode. If the file is successfully opened, the function reads all lines from the file. Each line is then split into columns based on commas to organize the data. The function proceeds to print a formatted table header for clarity. It iterates over each row of data, printing each row in a formatted manner to match the table structure. If the file is not found, the function catches the FileNotFoundError exception and prints an error message indicating that the file does not exist.

### read\_furniture\_products() function

The read\_furniture\_products function is designed to read and display the product details from inventory.txt. Similar to the previous function, it starts by attempting to open the file in read mode. Upon successful opening, it reads all lines from the file and splits each line into columns based on commas. The function then prints a formatted table header that includes columns for the product ID, name, quantity, and price. It iterates over each row of data, extracting the relevant columns (ID, product name, quantity, and price) and printing each row in a formatted manner. If the file is not found, the function catches the FileNotFoundError exception and prints an error message indicating the file does not exist.

### read\_inventory() function

The read\_inventory function reads the entire inventory from inventory.txt and returns the data as a list of lists. It attempts to open the file in read mode and read all lines from it. Each line is split into columns based on commas, and the resulting list of lists is returned. If the file is not found, the function catches the FileNotFoundError exception and prints an error message indicating the file does not exist.

### write\_inventory(inventory) function

The write\_inventory function writes the updated inventory data back to inventory.txt. It takes a list of lists representing the inventory as its input parameter. The function opens inventory.txt in write mode and iterates over each item in the inventory. Each item is converted to a comma-separated string and written to the file. This function ensures that the inventory file is updated with the latest data.

### generate\_invoice(transactions) function

The generate\_invoice() function generates an order invoice from a list of transaction details and saves it to a file. It creates a filename using the employee's name and the current date and time. The function prompts the user to input the shipping distance to determine the shipping cost, with options for "really far," "far," and "near," assigning respective costs of $100, $60, and $40. It then processes each transaction to extract details such as furniture ID, manufacturer, product name, quantity, price per unit, and total cost, accumulating these into lists and calculating the subtotal. The grand total is computed by adding the shipping cost to the subtotal. Finally, the order\_Invoice() function is called to generate the invoice file using the collected data and filename.

### add\_new\_product(furniture\_id, manufacturer, product\_name, quantity, price) function

The add\_new\_product function adds a new product to the inventory. It takes the product ID, manufacturer, product name, quantity, and price as input parameters. The function first reads the current inventory using read\_inventory(). It then appends the new product details to the inventory list. Finally, the function writes the updated inventory back to inventory.txt using the write\_inventory() function.

### update\_inventory(furniture\_id, quantity\_change) function

The update\_inventory function updates the quantity of a specific product in the inventory. It takes the product ID and the change in quantity (positive for adding stock, negative for reducing stock) as input parameters. The function reads the current inventory using read\_inventory(). It iterates over the inventory to find the item with the given product ID and updates its quantity accordingly. The updated inventory is then written back to inventory.txt using the write\_inventory() function.

### order\_furniture(furniture\_id, quantity, employee\_name) function

The order\_furniture() function handles the ordering of furniture items by updating the inventory and recording transaction details. It accepts a furniture ID, quantity, employee name, and a list of transactions.

First, the function checks if the item is in the inventory using read\_inventory(). If found, it updates the quantity with update\_inventory(), calculates the total cost, and adds the transaction details to the transactions list.

If the item is not found, it prompts the user to enter details for the new product and adds it to the inventory with add\_new\_product(). The function then recursively calls itself to process

### sell\_furniture(customer\_name, furniture\_id, quantity) function

The sell\_furniture() function processes sales transactions for furniture items, updating the inventory and generating a sales invoice. It begins by reading the current inventory and initializing variables for the subtotal, VAT amount, shipping cost, valid transactions, and invalid transactions. For each transaction, the function checks if the furniture item is available in the inventory and whether there is sufficient stock. If the item is available and in sufficient quantity, it calculates the total cost, updates the inventory, and appends the transaction to the list of valid transactions while updating the subtotal and VAT. If the item is not available or there is insufficient stock, the transaction is logged as invalid. After processing all transactions, the function prompts the user to input the customer's location to determine the shipping cost. It then calculates the grand total, including the shipping cost and VAT, and generates a sales invoice, saving it to a file named with the customer's name and the current date and time. If insufficient stock is available, an error message is printed.

# Pseudocode

Pseudocode is a full and easily comprehensible description of how exactly a certain program, or an algorithm should work. The script is articulated in more formal vocabulary but chosen deliberately to use the programming language’s natural structure and style, so that it can be more easily read and understood by programmers and other people involved in the project. Pseudocode is not a programming language, and one cannot compile the pseudocode to create an executable software. It only provides a skeleton for converting the logical code into a real programming language. (Sheldon, 2024)

## Pseudocode for main.py code

IMPORT read\_furniture\_data, read\_furniture\_products FROM read

IMPORT sell\_furniture, order\_furniture FROM operation

FUNCTION main()

WHILE True

OUTPUT "\n=============================================="

OUTPUT "BRJ Furniture Store Management System"

OUTPUT "=============================================="

OUTPUT "\n1. Display available furniture"

OUTPUT "2. Purchase furniture from manufacturer"

OUTPUT "3. Sell furniture to customer"

OUTPUT "4. Exit"

INPUT choice

IF choice == "1" THEN

CALL read\_furniture\_data()

END IF

ELSE IF choice == "2" THEN

WHILE True

OUTPUT "\nItem already available in furniture's inventory"

CALL read\_furniture\_products()

OUTPUT "If you want to buy the products which are not in inventory then enter the next ID"

INPUT furniture\_id

INPUT quantity

INPUT employee\_name

CALL order\_furniture(furniture\_id, quantity, employee\_name)

INPUT order\_more

IF order\_more != "yes" THEN

BREAK

END IF

END WHILE

END IF

ELSE IF choice == "3" THEN

WHILE True

INPUT customer\_name

CALL read\_furniture\_products()

INPUT furniture\_id

INPUT quantity

CALL sell\_furniture(customer\_name, furniture\_id, quantity)

INPUT sell\_more

IF sell\_more != "yes" THEN

BREAK

END IF

END WHILE

END IF

ELSE IF choice == "4" THEN

OUTPUT "-----------------------------------------"

OUTPUT "Thank you for choosing BRJ Furniture."

OUTPUT "-----------------------------------------"

BREAK

END IF

ELSE

OUTPUT "\nInvalid choice. Please try again."

END IF

END WHILE

END FUNCTION MAIN

## Pseudocode for read.py

FUNCTION read\_furniture\_data()

TRY

OPEN file "inventory.txt" FOR READING

READ all lines from the file into variable lines

SPLIT each line by comma and strip whitespace, store in variable data

OUTPUT header line: "\nID | Manufacturers | Product Name | Quantity | Price"

OUTPUT separator line: "-" \* 78

FOR each row in data

OUTPUT formatted row

OUTPUT separator line: "-" \* 78

END FOR

EXCEPT FileNotFoundError

OUTPUT "The file 'inventory.txt' does not exist."

END TRY

END FUNCTION

FUNCTION read\_furniture\_products()

TRY

OPEN file "inventory.txt" FOR READING

READ all lines from the file into variable lines

SPLIT each line by comma and space, store in variable data

OUTPUT header line: "ID | Product Name | Quantity | Price"

OUTPUT separator line: "-" \* 35

FOR each row in data

SET furniture\_id to row[0]

SET product\_name to row[2]

SET quantity to row[3]

SET price to row[4]

OUTPUT formatted row

END FOR

EXCEPT FileNotFoundError

OUTPUT "The file 'inventory.txt' does not exist."

END TRY

END FUNCTION

FUNCTION read\_inventory()

TRY

OPEN file "inventory.txt" FOR READING

READ all lines from the file into variable lines

RETURN lines split by comma and space, stripped of whitespace

EXCEPT FileNotFoundError

OUTPUT "The file 'inventory.txt' does not exist."

END TRY

END FUNCTION

## Pseudocode for write.py

IMPORT read\_inventory from read

FUNCTION write\_inventory(inventory)

OPEN file "inventory.txt" FOR WRITING

FOR each item in inventory

JOIN item elements with ", " and write to file

WRITE newline character to file

END FOR

CLOSE file

END FUNCTION

FUNCTION add\_new\_product(furniture\_id, manufacturer, product\_name, quantity, price)

CALL read\_inventory() TO GET inventory

APPEND new item [furniture\_id, manufacturer, product\_name, quantity as string, price] TO inventory

CALL write\_inventory(inventory)

END FUNCTION

## Pseudocode for operation.py

IMPORT datetime

IMPORT write\_inventory, add\_new\_product from write

IMPORT read\_inventory from read

FUNCTION update\_inventory(furniture\_id, quantity\_change)

CALL read\_inventory() TO GET inventory

FOR each item in inventory

IF item[0] equals furniture\_id THEN

GET current\_quantity FROM item[3] AS integer

CALCULATE new\_quantity = current\_quantity + quantity\_change

SET item[3] TO new\_quantity AS string

BREAK the loop

END IF

END FOR

CALL write\_inventory(inventory)

END FUNCTION

FUNCTION order\_furniture(furniture\_id, quantity, employee\_name)

CALL read\_inventory() TO GET product\_list

SET product\_found TO False

FOR each item in product\_list

IF item[0] equals furniture\_id THEN

SET product\_found TO True

CALL update\_inventory(furniture\_id, quantity)

GET price\_per\_unit FROM item[4] AS float (remove '$')

CALCULATE total\_cost = quantity \* price\_per\_unit

OPEN file 'order\_invoice\_of\_{employee\_name}\_{current\_date\_and\_time}.txt' FOR WRITING

WRITE invoice details to file

CLOSE file

BREAK the loop

END IF

END FOR

IF product\_found IS False THEN

PRINT message about adding new product

PROMPT user for manufacturer, product\_name, and price

CALL add\_new\_product(furniture\_id, manufacturer, product\_name, quantity, price)

CALL order\_furniture(furniture\_id, quantity, employee\_name) AGAIN

END IF

END FUNCTION

FUNCTION sell\_furniture(customer\_name, furniture\_id, quantity)

CALL read\_inventory() TO GET inventory

FOR each item in inventory

IF item[0] equals furniture\_id THEN

GET price\_per\_unit FROM item[4] AS float (remove '$')

CALCULATE total\_cost = quantity \* price\_per\_unit

CALCULATE vat\_amount = total\_cost \* 0.13

PRINT message about potential shipping cost

PROMPT user for location

IF location equals "2" THEN

SET shipping\_cost TO 50

ELSE

SET shipping\_cost TO 0

END IF

CALCULATE total\_amount = total\_cost + vat\_amount + shipping\_cost

IF quantity IS less THAN or equal to item[3] THEN

CALL update\_inventory(furniture\_id, -quantity)

OPEN file 'sale\_invoice\_of\_{customer\_name}\_{current\_date\_and\_time}.txt' FOR WRITING

WRITE invoice details to file

CLOSE file

BREAK the loop

ELSE

PRINT insufficient stock message

END IF

END IF

END FOR

END FUNCTION

# Flowchart

# Data Structures

In developing the furniture inventory management system, several data structures in Python are utilized to efficiently manage data storage, input/output operations, and transaction processing:

**Lists**: Used to store the list of furniture items read from and written to the text file. Lists allow for sequential access and modification of data entries, making them suitable for managing dynamic inventory updates and transaction histories.

**Dictionaries**: Employed to map unique identifiers (IDs) of furniture items to detailed information such as manufacturer, product name, available quantity, and price per unit. This facilitates quick look up and retrieval of specific item details during transaction processing and inventory updates.

**Strings**: Utilized for processing and formatting textual data, such as parsing input from administrators or formatting output for transaction receipts. String manipulation functions are employed to ensure data integrity and clarity in generated invoices and transaction notes.

**File Handling**: Python's built-in file handling capabilities are employed for input/output operations, allowing the program to read furniture data from and write updated data back to the text file. This ensures persistent storage and retrieval of inventory information between program sessions.

**Custom Class**: For more complex applications, custom-defined classes can encapsulate furniture item attributes and methods, enhancing code modularity and scalability. Classes can represent furniture items as objects with properties and behaviors tailored to specific application requirements.

Each data structure is chosen based on its suitability for handling various aspects of the inventory management system, from efficient data retrieval and manipulation to ensuring robust file handling and user interaction. (Akash, 2024)

# Program

# Testing

# Conclusion

In conclusion, the construction of the BRJ Furniture Store Inventory Management System has therefore been a worthy exercise in learning Python programming and data processing. It was somewhat difficult but also very rewarding to produce a functional approach to organizing stock, take payments, and issue receipts.  
  
I maintain clean and organized code that is written in the standard manner of software development. I was able to solve problems related to organizing and processing the data of inventory with the assistance of lists and dictionaries. Since the program is capable of reading from and writing text files, data is secured and retrievable for regular use.

I also learned skills in the design of algorithms, error management and file management from this project. I also got to know about the documentation of my work and it’s testing as well for it to be confirmed that it works effectively.  
  
During planning, coding, and debugging stages I obtained the experience in such aspects as team cooperation, critical thinking, and practical use of the learned material.  
  
Therefore, the BRJ Furniture Store Inventory Management System proves my commitment and IT competency. I am glad that I accomplished such things, and I think it will be useful in the store. General, this project has prepared me for future software development related tasks, and I will use the skills acquired in my future learning and working endeavors.

# References

Akash. (2024, July 17). *data-structures-in-python*. Retrieved from edureka: https://www.edureka.co/blog/data-structures-in-python/

Geeks For Geeks. (2024, July 16). *introduction-to-python*. Retrieved from Geeks For Geeks: https://www.geeksforgeeks.org/introduction-to-python/

Geeks for geeks. (2024, July 17). *python-features*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/python-features/

Nikolopoulou, K. (2024, July 28). *what-is-an-algorithm*. Retrieved from scribbr: https://www.scribbr.com/ai-tools/what-is-an-algorithm/

Python Software Foundation. (2024, July 16). *idle.html*. Retrieved from docs.Python: https://docs.python.org/3/library/idle.html

Sheldon, R. (2024, July 28). *pseudocode*. Retrieved from techtarget: https://www.techtarget.com/whatis/definition/pseudocode

# Appendix