



CS4051NI/CC4059NI Fundamentals of Computing

60% Individual Coursework

2023 Spring

Student Name: Saroj Babu Karki

London Met ID: 22067792

College ID: NP01CP4A220180

Assignment Due Date: Friday, July 28, 2023

Assignment Submission Date: Friday, July 28, 2023

Project File Links:

Google Drive Link:	https://drive.google.com/drive/folders/1bsXVrfHM8PxK78W2WkJ5caM_gEUUz4rN?usp=drive_link
--------------------	---

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.

Table of Contents

1.0	Introduction.....	1
1.1	Brief Introduction about the project.....	1
1.2	Goals and Objectives.....	2
2.0	Discussion and Analysis.....	3
2.1	Algorithm	3
2.1.1	Algorithm for laptop billing system.....	4
2.2	Flowchart	6
2.3	Pseudocode.....	7
2.3.1	Pseudocode for main.py module.....	7
2.3.2	Pseudocode for operation.py module.....	10
2.3.3	Pseudocode for read.py module	13
2.3.4	Pseudocode for write.py module	15
2.4	Data Structures.....	17
2.4.1	Some common data structures used in python	17
2.4.2	Implementation of data structures while developing laptop management system.....	19
	21
3.0	Program	22
3.1	Process of conducting laptop order operation.....	24
3.1.1	Conducting order process	24
3.1.2	Creation of text file and write order invoice to the file	26
3.1.3	Termination of order process	27
3.2	Process of conducting sell operation.....	28
3.2.1	Conducting sell process.....	28
3.2.2	Creation of text file and write sell invoice to the file	30
3.2.3	Termination of sell process	31
4.0	Testing	32
4.1	Testing 1 : To check implementation and functionality of try, except.....	32
4.2	Testing 2 : To select Order and Sell option for the laptop	34
4.2.1	Testing 2.1 : To input non existing value in order laptop.....	34
4.2.2	Testing 2.2 : To input non existing value in sell laptop.....	36
4.2.2	Testing 2.3 : To input negative value in order laptop	38

4.2.2 Testing 2.4 : To input negative value in sell laptop	40
4.3 Testing 3 : To display file generation of order laptops(Order multiple laptops).....	42
4.4 Testing 4 : To display file generation of sell laptops(Sell multiple laptops)	47
4.5 Testing 5 : To update the quantity being deducted in a text file while ordering or selling laptop	52
5.0 Conclusion	55
6.0 References	56
6.0 Appendix	57
6.0.1 Appendix : Code for main.py module	57
6.0.2 Appendix : Code for operation.py module.....	65
6.0.3 Appendix : Code for read.py module	74
6.0.4 Appendix : Code for write.py module	83

Table of Figures

Figure 1 Use of string data structure in development of laptop management system ...	19
Figure 2 Validation of string value	19
Figure 3 Use of string data structure to display invoice generator header.....	20
Figure 4 Use of string data structure for header of invoice.....	20
Figure 5 Declaring empty laptop list to store laptop details	21
Figure 6 Declaring value in list to store data in the list for sell operation	21
Figure 7 Test 1 : Use of try except block to validate input	32
Figure 8 Test 2 : Displaying error message for wrong value	33
Figure 9 Testing 2.1 : Displaying error message for index which is not available during laptop order process.....	35
Figure 10 Testing 2.2 : Displaying error message for index which is not available during laptop sell process.....	37
Figure 11 Testing 2.3 : Displaying error message if index is negative for order process	39
Figure 12 Testing 2.4 : Displaying error message if index is negative for sell process .	41
Figure 13 Testing 3 : Conducting order process and displaying invoice on screen.....	45
Figure 14 Testing 3 : Allocation of text file after order invoice is generated and written to text file.....	46
Figure 15 Testing 3 : Written order invoice in text file	46
Figure 16 Testing 4 : Conducting sell process and displaying sell invoice on the screen	50
Figure 17 Testing 4 : Allocating sell invoice text file after generating and displaying sell invoice on screen	51
Figure 18 Testing 4 : Written sell invoice to a text file	51
Figure 19 Testing 5 : Displaying Laptop quantity update on screen for order operation	53
Figure 20 Testing 5 : Displaying Laptop quantity update on text file for order operation	53
Figure 21 Testing 5 : Displaying Laptop quantity update on screen for sell operation ..	54
Figure 22 Testing 5 : Displaying Laptop quantity update on text file for sell operation ..	54

Table of tables

Table 1 Testing 1 : To check implementation and functionality of try, except.....	32
Table 2 Testing 2.1 : To input non existing value in order laptop	34
Table 3 Testing 2.2 : To input non existing value in sell laptop	36
Table 4 Testing 2.3 : To input negative value in order laptop	38
Table 5 4.2.2 Testing 2.4 : To input negative value in sell laptop	40
Table 6 Testing 3 : To display file generation of order laptops(Order multiple laptops). 43	
Table 7 Testing 4 : To display file generation of sell laptops(Sell multiple laptops)	48
Table 8 4.5 Testing 5 : To update the quantity being deducted in a text file while ordering or selling laptop	52

1.0 Introduction

1.1 Brief Introduction about the project

Python is a high-level, general-purpose programming language that is widely used. Python programming language is utilized in web development, machine learning applications, and other software industry technology.

Python is now the most extensively used multi-purpose, high-level programming language, allowing programming in both the Object-Oriented and Procedural paradigms. Python applications are typically smaller than programs written in other programming languages such as Java or other programming languages. Programmers must type relatively little, and the indentation requirement of the language ensures that their code is always readable and easy to detect errors and bugs. (Anon., 2023)

In today's evolving world which is fast paced, dealing business related to laptop buying and rentals have importance to manage the operations of business transactions in a laptop store. Billing is necessary to track the necessary records to evaluate profit and loss. Manual billing methods are also possible, but it is time consuming and unreliable. At the same time it creates a challenge in handling a large scale transaction with lots of errors which is difficult to detect fast and difficult to solve the issue. To address all these difficulties and challenges we have proposed a plan to develop a laptop billing system using Python programming language.

The project aims in providing a good and effective solution to all the difficulties and challenges that are faced by the laptop store while conducting transactions in a business. The system is designed on the basis of easy use and accessibility, reliability and custom formatting of the system. The tools are utilized in this designing and development project using Python programming. It provides required insights and reports that help to improve services of the store as well as help in decision making for long term benefit in business operation.

Overall, developing a laptop billing system in Python is a wise investment for any company that deals with laptop rentals and repairs. It can assist firms in improving their

operations, increasing efficiency, and lowering expenses. In the following parts, we will go over the specifics of this project, such as its design, implementation, and testing.

1.2 Goals and Objectives

The project have the following aims and objectives which are mentioned below.

- To allow programs to read text files containing information about available laptops in the store.
- To update laptop data in real time.
- To write easy-to-use programs that can handle different types of transactions, as per the need of user
- To order laptops from manufacturers directly and sell laptops to customers.
- Create a Text file or invoice for each transaction with all required details.
- Allow the program to automatically update the laptop quantity in the text file each time when order or sell operation is conducted.
- Display transaction details that reflects the current stock of each laptop.

2.0 Discussion and Analysis

The construction of a laptop billing system using the Python programming language is a worthwhile investment for organizations that deal with laptop rentals and repairs. In this section, we will explore and analyse the important aspects of this project, such as its design, implementation, and testing. While developing the system a particular flow is maintained during this process. Algorithm is developed to know what the code is going to be. Flowcharts are drawn to understand the data flow. After all a final python code is written according to the algorithm and flowchart using appropriate data structures.

The laptop billing system is implemented in various steps, including data modelling, coding, and testing. Python code has been built to handle the system's different components, such as inventory management, invoice generation, and report generation.

2.1 Algorithm

An algorithm is a process for solving a problem or completing a computation. Algorithms are specific lists of instructions that perform specified tasks step by step in either hardware or software-based processes.

Algorithms are frequently utilized in all fields of IT. In mathematics and computer science, an algorithm is a simple technique that solves a recurring problem. Algorithms are also employed as specifications for data processing and play an important part in automated systems.

Algorithms can be expressed using natural languages, computer languages, pseudocode, flowcharts, and control tables. Natural language expressions are uncommon because they are more ambiguous. Programming languages are typically used to express algorithms that are executed by a computer. (S.Gillis, 2022)

2.1.1 Algorithm for laptop billing system

Step 1 : Start

Step 2 : Display shop title, address, phone number and contact number along with greeting message

Step 3 : Display options from 1 to 4

- a) If option is 1, Order laptop
- b) If option is 2, Sell laptop
- c) If option is 3, Show laptop stock
- d) If option is 4, Exit

Step 4 : If option selection is 1, then go to **Step 5**

Step 5 : Display laptop details in a formatted table for order process.

Step 6 : Read index and quantity of laptop and update the quantity

Step 7 : Ask, if the user want to order more or not. If yes then go to **Step 6** otherwise go to **Step 8**

Step 8 : Display total amount and thank you message

Step 9 : Read distributor name and phone number

Step 10 : Read laptop brand, model, quantity and price

Step 11 : If user want to order more then go to Step 10 otherwise go to Step 12

Step 12 : Display invoice on the screen including gross amount, 13% VAT and Total amount after 13% VAT and write order invoice to a text file then go to **Step 13**

Step 13 : If option selection is 2 then go to **Step 14**

Step 14 : Display laptop table in formatted order and go to **Step 15**

Step 15 : Read Index and Quantity of laptop they want to process sell and update the quantity

Step 16 : Ask, if the user want to order more or not. If yes then go to **Step 15** otherwise go to **Step 17**

Step 17 : Display total amount and thank you message

Step 18 : Read customer name and phone number

Step 19 : Read laptop brand, model, quantity and price

Step 17 : Ask customer if they want to buy more laptop. If yes go to **sStep 19** otherwise go to **Step 20**

Step 20 : Ask customer if they want shipping. If yes add \$200 and go to **Step 21** otherwise go to **Step 22**

Step 21 : Display invoice for sell with gross amount, shipping cost \$200 and total amount.

Step 22 : Display invoice for sell with gross amount excluding shipping cost and display total amount and go to Step 3.

Step 23 : If option selection is 3, then go to Step 24

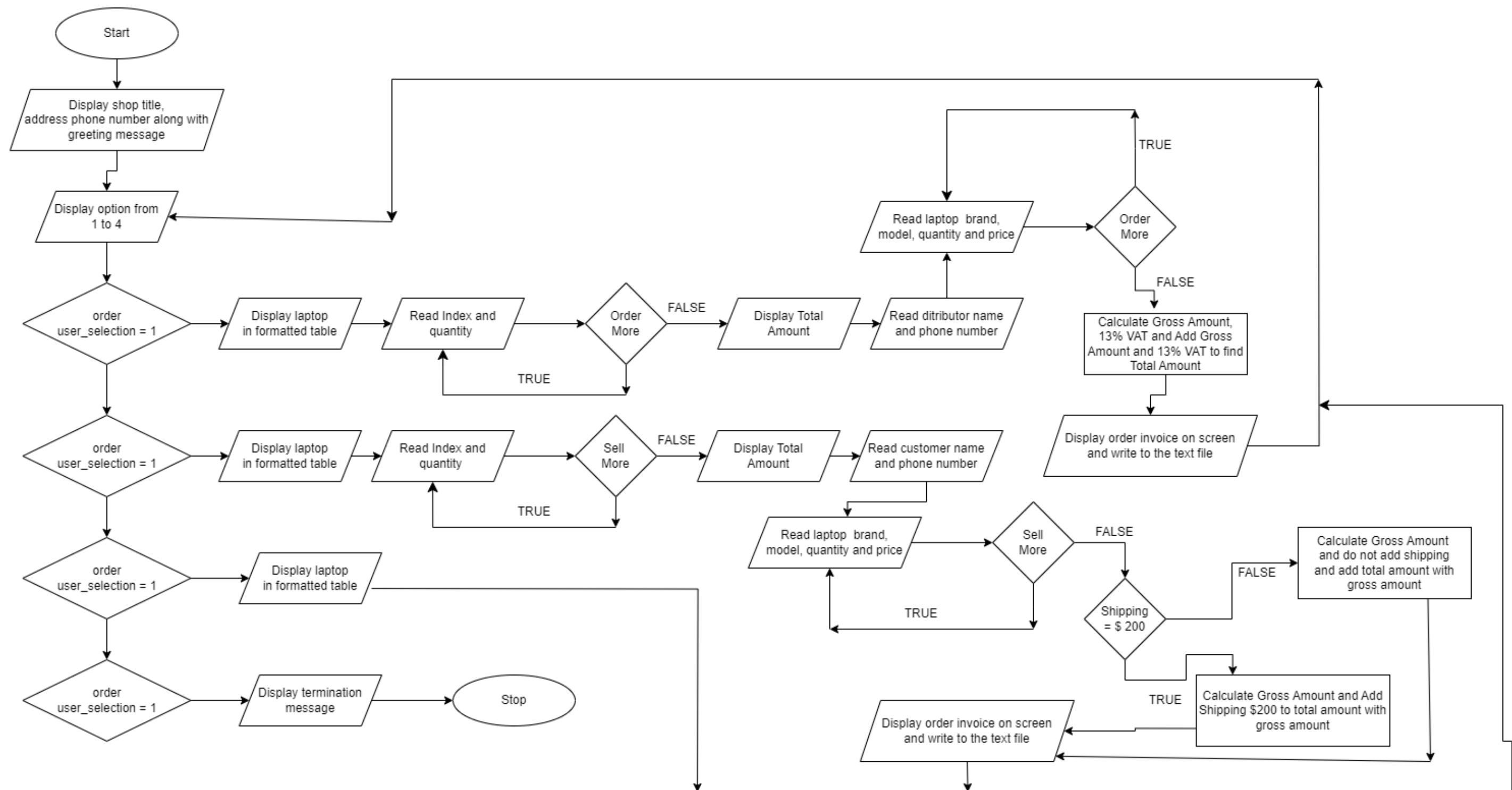
Step 24 : Display laptop table in formatted order and go to step 3

Step 25 : If option selection is 4, then go to Step 26

Step 26 : Stop

2.2 Flowchart

An illustration of a process or workflow's steps, sequences, and decisions is called a flowchart. Although there are other variations of flowcharts, a basic flowchart is the most straightforward representation of a process map. It is an effective tool for planning, visualising, recording, and improving processes in a variety of industries. (Anon., 2023)



2.3. Pseudocode

2.3.1 Pseudocode for main.py module

IMPORT read

IMPORT write

IMPORT operation

DECLARE a function **main_screen_display()**

DISPLAY title of the shop, address and contact information

DECLARE a function **validate_input_from_user()**

WHILE true

ASK an option to continue operation

IF the input is empty

DISPLAY an error message and continue

TRY to convert input into integer

IF input is not an integer

DISPLAY an error message

RETURN integer input

MAIN CODE

WHILE true

CALL **main_screen_display()**

WHILE true

DISPLAY options

 Option 1: Order Laptop

 Option 2: Sell Laptop

 Option 3: Check Laptop Stock

 Option 4: Exit

ASK to select an option using **validate_input_from_user()**

IF option is not from 1 to 4

DISPLAY error message and continue

IF option is 1

DISPLAY question asking which laptop the user wants to order

DISPLAY list of available laptops in a tabular format

CALL **read.display_laptop_list_table()** and **read.laptop_quantity_update()** to conduct order process

CALL **operation.get_order_calculation()** to get order calculation details

IF operation is successful (no validation errors)

CALL **write.generate_and_display_order_invoice()** and display invoice

CALL **write.write_order_invoice()** to write order invoice to a text file

IF option is 2

DISPLAY question asking which laptop the user wants to sell

DISPLAY list of available laptops in a tabular format

CALL **read.display_laptop_list_table()** and **read.laptop_quantity_update()** to conduct sell process

CALL **operation.get_sell_calculation()** to get sell calculation details

IF operation is successful (no validation errors)

CALL **write.generate_and_display_sell_invoice()** to display sell invoice

CALL write.write_sell_invoice() to write the sell invoice to a text file

IF option is 3

DISPLAY available laptops and their stock in a tabular format

IF option is 4

DISPLAY system has been terminated message

EXIT the program

2.3.2 Pseudocode for operation.py module

DECLARE a function “**input_validation**”

WHILE true

ASK user to input value with this prompt

IF input is empty

DISPLAY error message

IF expected data type is string but, input is numeric

DISPLAY error message and loop back

IF expected data type is integer

IF input is not numeric

DISPLAY error message and loop back

CONVERT input as integer before returning

IF expected data type is Boolean

IF input is not “yes” or “no”

DISPLAY error message and loop back

CONVERT input as Boolean before returning

RETURN input value as it is

DECLARE a function “**get_order_calculation()**”

CALL FUNCTION “**get_distributor_laptop_order_details()**”

DECLARE a function “**get_sell_calculation()**”

CALL FUNCTION “**get_customer_laptop_sell_details()**”

DECLARE a function “**get_distributor_laptop_order_details()**”

ASK distributor name using **input_validation()**

ASK distributor phone number using **input_validation()**

CREATE an empty list

WHILE true

ASK laptop brand using **input_validation()**

ASK laptop model using **input_validation()**

ASK laptop quantity using **input_validation()**

ASK laptop price using **input_validation()**

CALCULATE gross amount of the laptop

CALCULATE VAT amount (13% of gross amount)

CALCULATE total amount

APPEND "laptop_details" to "laptop_list"

ASK if they want to order more laptops using **input_validation()**

IF "yes"

CONTINUE

ELSE

BREAK

CALCULATE total invoice amount by summing up the total amounts of each laptop in 'laptop_list'

CALCULATE total gross amount by summing up the gross amounts of each laptop in 'laptop_list'

RETURN distributor_name, laptop_list, distributor_laptop_gross_amount, distributor_total_invoice_amount, distributor_phone_no

CALL get_order_calculation()

DECLARE a function get_customer_laptop_sell_details()

ASK customer name using **input_validation()**

ASK customer phone number using **input_validation()**

CREATE an empty list **laptop_list**

WHILE true

ASK laptop brand using **input_validation()**

ASK laptop model using **input_validation()**

ASK laptop quantity using **input_validation()**

ASK laptop price using **input_validation()**

CALCULATE gross amount of laptop

APPEND laptop details to "**laptop_list**"

ASK IF user want to purchase more laptops using **input_validation()**

IF "yes"

CONTINUE

ELSE

BREAK

CALCULATE total invoice amount by adding the gross amounts of each laptop in '**laptop_list**'

ASSUME initial shipping as '0'

RETURN sell_customer_name, laptop_list, sell_total_invoice_amount,
sell_phone_no, sell_shipping_cost

CALL get_sell_calculation()

2.3.3 Pseudocode for read.py module

DECLARE a text file and assign "laptop info.txt"

OPEN the text file "**laptop info.txt**" in reading mode and assign it to "**laptop_details_file**"

STORE laptop details in **laptop_list**

DECLARE a function to display the list in formatted table

DEFINE header list for the table of laptop details

DISPLAY formatted table with proper formatting

FOR each laptop in "**laptop_list**"

DISPLAY each row in formatted table

DECLARE a function to update quantity of laptop

WHILE true

ASK user to enter laptop index

IF input is empty, not a numeric value

DISPLAY error message and loop back to ask laptop index

CONVERT the input to integer and check if it is valid index or not

IF the index is invalid

DISPLAY error message and loop back

BREAK loop if index is valid

WHILE true

ASK user to enter quantity of laptop

IF input is empty, not a numeric value

DISPLAY error message and loop back to ask laptop quantity

CONVERT the input to integer and check if it is valid index or not

IF the quantity is invalid

DISPLAY error message and loop back

BREAK loop if quantity is valid

UPDATE the quantity of laptop in “**laptop_list**”

OPEN “**laptop info.txt**” file in write mode assign to “**laptop_details_file**”

WRITE quantity and other required details in “**laptop info.txt**” file

CALCULATE total amount based on reduced quantity by user

DISPLAY total amount

WHILE true

ASK if user want to buy more laptop

IF input is empty, not a numeric value

DISPLAY error message

IF “yes”

CALL update quantity

ELSE IF “no”

DISPLAY “Thank you message” and exit the code

CALL update quantity function() and initiate the update process

2.3.4 Pseudocode for write.py module

IMPORT datetime

GET current date and time

GENERATE formatted date and time in string

EXTRACT day, month, year and hour, minute and second

CREATE a string in formatted order to extract values in dd/mm/yy hh:mm:ss format

DECLARE a function to generate and display order invoice

DISPLAY header and include distributor details, date and time

IF the list **laptop_list** is not empty

DISPLAY laptop details in a formatted table

DISPLAY gross amount, 13% VAT amount and total amount including 13% VAT

DISPLAY "Thank you for shopping with us !!!" message

DISPLAY "Visit Again !!!" message

DECLARE a function to write the order invoice to a text file

CREATE a "filename" name assigned to the distributor in text file

OPEN file with "filename" in write mode and assign the "file"

WRITE header and include distributor details, date and time

IF the list **laptop_list** is not empty

WRITE laptop details in a formatted table

WRITE gross amount, 13% VAT amount and total amount including 13% VAT

WRITE "Thank you for shopping with us !!!" message

WRITE "Visit Again !!!" message

DECLARE a function to generate and display sell invoice

DISPLAY header and include customer details, date and time

IF the list **laptop_list** is not empty

DISPLAY laptop details in a formatted table

DISPLAY gross amount, Shipping Cost and total amount including Shipping Cost

DISPLAY "Thank you for shopping with us !!!" message

DISPLAY "Visit Again !!!" message

DECLARE a function to write display sell invoice

CREATE a "filename" name assigned to the customer in text file

OPEN file with "filename" in write mode and assign the "file"

WRITE header and include customer details, date and time

IF the list **laptop_list** is not empty

WRITE laptop details in a formatted table

WRITE gross amount, Shipping Cost and total amount including Shipping Cost

WRITE "Thank you for shopping with us !!!" message

WRITE "Visit Again !!!" message

2.4 Data Structures

Data structures are a means to arrange data such that, depending on the circumstance, it can be retrieved more quickly. Any programming language's essential building block upon which a programme is based is the data structure. Compared to other programming languages, Python makes it easier to master the fundamentals of these data structures. (Anon., 2023)

2.4.1 Some common data structures used in python

List :

Lists are used to sequentially store data of various data kinds. Every item in the list, also known as the Index, has an address allocated to it. The index value begins at 0 and continues until the final component, which is the positive index. You can also retrieve elements from last to first using negative indexing, which starts at -1. (Anon., 2023)

For Example

```
Created_List_ = [ 1, 3, 5, "Hello", 2.5]
```

The list contains integer, string and float values.

Tuple:

The only difference between tuples and lists is that once the data has been entered, it cannot be modified under any circumstances. The only exception is if the tuple's data is mutable, in which case it can be altered. You can better comprehend with the aid of the sample programme. (Anon., 2023)

For Example

```
Created_tuple = (1, 2, 3, 4, "Mahesh", 5.0 )
```

The tuple contains integer, string and float values.

Dictionary:

In contrast to other Data Types, which can only retain a single value as an element, a Python dictionary can hold both the key and value. It is used to store data values in curly bracket {}. The dictionary includes key-value pairs to help it become more efficient.

With the use of keys, the Python Dictionary is indexed. They can be of any hash type, i.e., a constant object like a text, integer, tuple, etc. Using curly brackets {} or dictionary comprehension, we can build a dictionary. (Anon., 2023)

For Example:

```
Dict = { "name": "John", "age":23 }
```

There are key-value pairs in this dictionary, where the keys are strings and the values might be of any data type.

Sets:

Sets are collections of distinct, unordered components. The data would only be added into the set once, even if it were repeated more than once. It is similar to the sets you have studied in mathematics. The operations are identical to how they are with arithmetic sets. (Anon., 2023)

For Example

```
My_set_num = {1, 2, 3}
```

The set allows integer value only and do not duplicate the data.

String:

Python Strings are collections of bytes that correspond to Unicode characters. A string is an immutable collection of characters, to put it simply. A single character in Python is just a string with a length of 1, as the language lacks a character data type.

For Example

String = "Hello world"

The string hold the characters "Hello world".

2.4.2 Implementation of data structures while developing laptop management system

While developing this laptop management system, I have used various types of data structures in this project. Each data structure have its own use and functionality.

In most cases, I have used strings to display lines of code. Those strings were not initialized but set to default. Those string values were displayed while asking a prompt to the user or displaying a statement or message to the user. To display a unique message and ask unique prompt to the user, I have initialized it with different names. Each string is not changeable and displays appropriate message and result when a data is entered in the field where strings are added.

```
def main_screen_display():
    # Display the title, address and contact details of the shop and greet the user
    print(".....")
    print("")
    print("                KARKI ELECTRONICS AND LAPTOP SUPPLIERS                ")
    print("                KAPAN, KATHMANDU | CONTACT: 01-4270603                ")
    print("")
    print(".....")
    print("                WELCOME TO KARKI ELECTRONICS AND LAPTOP SUPPLIERS...                ")
    print("")
    print(".....")
```

Figure 1 Use of string data structure in development of laptop management system

```
# A list is created to store order details.
laptop_list = []

while True:
    distributor_laptop_brand_name = input_validation("    ENTER LAPTOP BRAND: ",str)
    distributor_laptop_model_name = input_validation("    ENTER LAPTOP MODEL: ",str)
    distributor_laptop_quantity = input_validation("    ENTER LAPTOP QUANTITY: ",int)
    distributor_laptop_price = input_validation("    ENTER LAPTOP PRICE: $",int)
```

Figure 2 Validation of string value


```
print("")
print("=====")
print("                TO GENERATE ORDER INVOICE PLEASE ENTER THE FOLLOWING DETAILS")
print("=====")
print("")
```

Figure 3 Use of string data structure to display invoice generator header

```
# Function to write order invoice to .txt file
def write_order_invoice(distributor_name, laptop_list, distributor_laptop_gross_amount, distributor_total_invoice_amount, distributor_phone_no):
    filename = f"{distributor_name}_order_invoice.txt"
    with open(filename, "w") as file:
        file.write("=====")
        file.write("")
        file.write("                KARKI ELECTRONICS AND LAPTOP SUPPLIERS")
        file.write("                ")
        file.write("                KAPAN, KATHMANDU | CONTACT: 01-4270603")
        file.write("                ")
        file.write("")
        file.write("                ORDER INVOICE")
        file.write("=====")
        file.write("Name of customer: " + str(distributor_name) + "\n")
        file.write("Phone no: " + str(distributor_phone_no) + "\n")
        file.write("Date: " + str(string_date_time + "\n"))
        file.write("=====")

    if laptop_list:
        file.write("-----")
        file.write("{:<15}{:<15}{:<15}{:<15}{:<15}".format("LAPTOP BRAND", "LAPTOP MODEL", "QUANTITY", "UNIT PRICE", "AMOUNT"))
        file.write("-----")
        for laptop in laptop_list:
            file.write("{:<15}{:<15}{:<15}{:<15}{:<15}".format(laptop[0], laptop[1], laptop[2], "$" + str(laptop[3]), "$" + str(laptop[4])))

        file.write("-----")
        file.write("{:<30}          Gross Amount:      ${}\n".format("", distributor_laptop_gross_amount))
        file.write("{:<30}          13% VAT:          ${}\n".format("", sum(product[5] for product in laptop_list)))
        file.write("{:<30}          Total Amount:      ${}\n".format("", distributor_total_invoice_amount))
        file.write("=====")
        file.write("")
        file.write("                THANK YOU FOR SHOPPING WITH US !!!")
        file.write("                VISIT AGAIN !!!")
        file.write("")
        file.write("=====")
```

Figure 4 Use of string data structure for header of invoice

Apart from the string, the data type that I used in my project is list. Instead of list, dictionary can possibly be used but in this scenario it was not possible. Dictionary have a unique key values. Each key with unique value was difficult to assign to the laptop since large amount of data was being used in the system.

It was much more convenient and effective to use list data structure to store data and append it. To store the laptop order and sell details it was useful.

Mostly, list is generally divided to two types, 1D list and 2D list. These list have different functions. In 1D list I was able to access data in one dimension. But, 2D list allowed me to access data in two dimensions as well. The list is used to store infinite amount of data without any hassle.

```

# A list is created to store order details.
laptop_list = []

while True:
    distributor_laptop_brand_name = input_validation("    ENTER LAPTOP BRAND: ",str)
    distributor_laptop_model_name = input_validation("    ENTER LAPTOP MODEL: ",str)
    distributor_laptop_quantity = input_validation("    ENTER LAPTOP QUANTITY: ",int)
    distributor_laptop_price = input_validation("    ENTER LAPTOP PRICE: $",int)

```

Figure 5 Declaring empty laptop list to store laptop details

```

# 13% VAT amount is calculated from the gross amount
distributor_VAT_amount = 0.13 * distributor_laptop_gross_amount

# Calculate total amount
distributor_total_amount = distributor_laptop_gross_amount + distributor_VAT_amount

# Add laptop details to the list created.
laptop_list.append([distributor_laptop_brand_name,
                    distributor_laptop_model_name,
                    distributor_laptop_quantity,
                    distributor_laptop_price,
                    distributor_laptop_gross_amount,
                    distributor_VAT_amount,
                    distributor_total_amount])

"""
If the user want to order more loop the prompt
otherwise display the invoice.
"""
distributor_order_more_laptop = input_validation("    DO YOU WANT TO ORDER MORE LAPTOPS? (YES/NO): ",bool)

sell_customer_name = input_validation("    ENTER CUSTOMER NAME: ", str)
sell_phone_no = input_validation("    ENTER PHONE NUMBER: ", int)

# Store the product sold to the customer.
laptop_list = []

while True:
    sell_laptop_brand_name = input_validation("    ENTER LAPTOP BRAND: ", str)
    sell_laptop_model_name = input_validation("    ENTER LAPTOP MODEL: ", str)
    sell_laptop_quantity = input_validation("    ENTER LAPTOP QUANTITY: ", int)
    sell_laptop_price = input_validation("    ENTER LAPTOP PRICE: $", int) # Use float to handle decimal prices

    sell_gross_amount = sell_laptop_quantity * sell_laptop_price

    laptop_list.append([sell_laptop_brand_name,
                        sell_laptop_model_name,
                        sell_laptop_quantity,
                        sell_laptop_price,
                        sell_gross_amount])

```

Figure 6 Declaring value in list to store data in the list for sell operation

3.0 Program

The project in which I have enrolled myself is focused on facilitating order and sell process of laptop to the customer which is completely based on using file handling concept in python programming language.

Generally, this program code developed is divided into four modules `main.py`, `operation.py`, `read.py` and `write.py`. Each modules have their own function and speciality. The program work in smooth manner and provide good user experience.

The **`main.py`** module is heart of other module. This module displays the screen to conduct the operation process. A greeting message and options are displayed. Four options are available to the user. It calls from other module to work properly. When user enters 1, 2 and 3 a table is displayed which includes the details about the laptop. 1 is for ordering the laptop by getting the details from the distributor and display order invoice. 2 is for selling laptop by getting details from the customer. 3 is to display current or updated details and 4 is to close the system. When the user enters any wrong value in the prompt an error message is displayed and looped back to the prompt.

The **`operation.py`** module conducts operation process of the laptop system. This module validates the input prompt and details entered by the user and displays suitable result. If the user enters any wrong input or leave the field empty an error message is displayed on the screen. Order and Sell details are taken from the user. In order function the distributor details are taken and amount is calculated with 13% VAT. In sell function the customer details are taken amount is calculated with shipping charge which is optional and call `write.py` module to display the invoice. Multiple products can also be ordered or sold by asking the distributor or customer.

The **`read.py`** module reads the text file "**`laptop info.txt`**" displays the laptop details in a formatted table. After, the table is displayed the user have to enter the index and quantity of the laptop they want to buy. If the user input is empty or invalid a message with error

is displayed. Multiple laptop can also be added by asking the user. If the user don't want to add more laptops an update message is displayed and total amount is displayed on the screen.

The **write.py** module displays the invoice with all the order details and sell details after taking input from **operation.py** module. Invoices are generated in real date and time. After invoice is generated for order or sell, the invoice is written to a text file with customer name along with whether the invoice is order invoice or sell invoice.

With the combination of all four modules **main.py**, **operation.py**, **read.py**, and **write.py** a final program for laptop management system is developed with all the necessary operations required. These combination of modules make the program easy to run and make necessary adjustment. When any task related to order or sell is to be conducted this program lets the user to make easy and reliable source to conduct day to day task for recording and conducting order and sell transaction.

3.1 Process of conducting laptop order operation

3.1.1 Conducting order process

KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603

WELCOME TO KARKI ELECTRONICS AND LAPTOP SUPPLIERS...

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM

ENTER YOUR SELECTION TO CONTINUE OPERATION: 1

WHICH LAPTOP DO YOU WANT TO ORDER ?

S.N	MODEL	BRAND	PRICE	QUANTITY	PROCESSOR	GRAPHICS
1	Razer Blade	Razer	\$2000.0	451	i7 7th Gen	GTX 3060
2	XPS	Dell	\$1976.0	956	i5 9th Gen	GTX 3070
3	Alienware	Alienware	\$1978.0	230	i5 9th Gen	GTX 3070
4	Swift 7	Acer	\$900.0	650	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	\$3500.0	500	i5 9th Gen	GTX 3070

ENTER THE INDEX OF LAPTOP: 1
ENTER QUANTITY:3

LAPTOP QUNATITY UPDATED IN THE TEXT FILE.

TOTAL AMOUNT : \$6000.0

DO YOU WANT TO BUY MORE LAPTOPS? (YES/NO): yes

ENTER THE INDEX OF LAPTOP: 2
ENTER QUANTITY:5

LAPTOP QUNATITY UPDATED IN THE TEXT FILE.

TOTAL AMOUNT : \$9880.0

.....
DO YOU WANT TO BUY MORE LAPTOPS? (YES/NO): no
.....
.....

THANK YOU FOR SHOPPING WITH US !
VISIT AGAIN !!!
.....

=====

TO GENERATE ORDER INVOICE PLEASE ENTER THE FOLLOWING DETAILS

=====

ENTER DISTRIBUTOR NAME: Toshiba Laptop Store

ENTER PHONE NUMBER: 9845342180
ENTER LAPTOP BRAND: Razer
ENTER LAPTOP MODEL: Razer Blade
ENTER LAPTOP QUANTITY: 3
ENTER LAPTOP PRICE: \$2000
DO YOU WANT TO ORDER MORE LAPTOPS? (YES/NO): yes
ENTER LAPTOP BRAND: Dell
ENTER LAPTOP MODEL: XPS
ENTER LAPTOP QUANTITY: 5
ENTER LAPTOP PRICE: \$1976
DO YOU WANT TO ORDER MORE LAPTOPS? (YES/NO): no

=====

KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603

=====

=====

ORDER INVOICE

=====

Name of customer: Toshiba Laptop Store

Phone no: 9845342180

Date: 27/7/2023 20:13:16

=====

LAPTOP BRAND	LAPTOP MODEL	QUANTITY	UNIT PRICE	AMOUNT
Razer	Razer Blade	3	\$2000	\$6000
Dell	XPS	5	\$1976	\$9880

=====

Gross Amount: \$15880

13% VAT: \$2064.4

Total Amount: \$17944.4

=====

THANK YOU FOR SHOPPING WITH US !!!
VISIT AGAIN !!!
=====

3.1.2 Creation of text file and write order invoice to the file

Name	Date modified	Type	Size
__pycache__	7/27/2023 6:31 PM	File folder	
laptop info	7/27/2023 8:14 PM	Text Document	1 KB
main	7/27/2023 6:56 AM	Python Source File	11 KB
operation	7/27/2023 6:56 AM	Python Source File	11 KB
read	7/26/2023 2:32 PM	Python Source File	12 KB
Toshiba Laptop Store_order_invoice	7/27/2023 8:16 PM	Text Document	2 KB
write	7/27/2023 6:56 AM	Python Source File	13 KB

Discussion and Analysis
Toshiba Laptop Store_order_invoice

File Edit View

```

=====
                        KARKI ELECTRONICS AND LAPTOP SUPPLIERS
                        KAPAN, KATHMANDU | CONTACT: 01-4270603
=====
                        ORDER INVOICE
=====
Name of customer: Toshiba Laptop Store
Phone no: 9845342180
Date: 27/7/2023  20:13:16
=====
-----
LAPTOP BRAND   LAPTOP MODEL   QUANTITY   UNIT PRICE   AMOUNT
-----
Razer          Razer Blade    3          $2000        $6000
Dell           XPS            5          $1976        $9880
-----
                        Gross Amount:    $15880
                        13% VAT:        $2064.4
                        Total Amount:   $17944.4
=====
                        THANK YOU FOR SHOPPING WITH US !!!
                        VISIT AGAIN !!!
=====

```

Ln 7, Col 254
100%
Windows (CRLF)
UTF-8

3.1.3 Termination of order process

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP

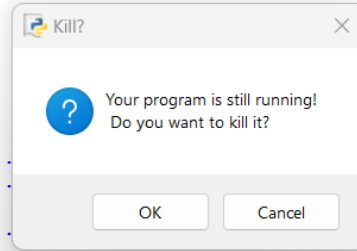
PRESS 2 : TO SELL LAPTOP

PRESS 3 : TO CHECK LAPTOP STOCK

PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM

ENTER YOUR SELECTION TO CONTINUE OPERATION: 4

THE SYSTEM HAS BEEN TERMINATED.



3.2 Process of conducting sell operation

3.2.1 Conducting sell process

```

.....
KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603
.....

WELCOME TO KARKI ELECTRONICS AND LAPTOP SUPPLIERS...
.....

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM
.....

ENTER YOUR SELECTION TO CONTINUE OPERATION: 2
.....

WHICH LAPTOP DO YOU WANT TO SELL?
.....

+-----+-----+-----+-----+-----+-----+-----+
| S.N | MODEL | BRAND | PRICE | QUANTITY | PROCESSOR | GRAPHICS |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Razer Blade | Razer | $2000.0 | 448 | i7 7th Gen | GTX 3060 |
| 2 | XPS | Dell | $1976.0 | 951 | i5 9th Gen | GTX 3070 |
| 3 | Alienware | Alienware | $1978.0 | 230 | i5 9th Gen | GTX 3070 |
| 4 | Swift 7 | Acer | $900.0 | 650 | i5 9th Gen | GTX 3070 |
| 5 | Macbook Pro 16 | Apple | $3500.0 | 500 | i5 9th Gen | GTX 3070 |
+-----+-----+-----+-----+-----+-----+-----+

ENTER THE INDEX OF LAPTOP: 3
ENTER QUANTITY:6
.....

LAPTOP QUNATITY UPDATED IN THE TEXT FILE.
.....

TOTAL AMOUNT : $11868.0
.....

DO YOU WANT TO BUY MORE LAPTOPS? (YES/NO): yes
.....

ENTER THE INDEX OF LAPTOP: 5
ENTER QUANTITY:20
.....

LAPTOP QUNATITY UPDATED IN THE TEXT FILE.
.....

TOTAL AMOUNT : $70000.0
.....

DO YOU WANT TO BUY MORE LAPTOPS? (YES/NO): no
.....

```

THANK YOU FOR SHOPPING WITH US !
VISIT AGAIN !!!

TO GENERATE ORDER INVOICE PLEASE ENTER THE FOLLOWING DETAILS

ENTER CUSTOMER NAME: Alex Gustavo Costa
ENTER PHONE NUMBER: 9988234456
ENTER LAPTOP BRAND: Alienware
ENTER LAPTOP MODEL: Alienware
ENTER LAPTOP QUANTITY: 6
ENTER LAPTOP PRICE: \$1978
DO YOU WANT TO PURCHASE MORE LAPTOPS? (YES/NO): yes
ENTER LAPTOP BRAND: Apple
ENTER LAPTOP MODEL: Macbook Pro 16
ENTER LAPTOP QUANTITY: 20
ENTER LAPTOP PRICE: \$3500
DO YOU WANT TO PURCHASE MORE LAPTOPS? (YES/NO): no
DO YOU WANT SHIPPING (YES/NO): yes

KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603

SELL INVOICE

CUSTOMER NAME: Alex Gustavo Costa
PHONE NUMBER: 9988234456
Date: 27/7/2023 21:52:11

LAPTOP BRAND	LAPTOP MODEL	QUANTITY	UNIT PRICE	AMOUNT
Alienware	Alienware	6	\$1978	\$11868
Apple	Macbook Pro 16	20	\$3500	\$70000

GROSS AMOUNT: \$82068
SHIPPING COST: \$200

TOTAL: \$82268

THANK YOU FOR SHOPPING WITH US !!!
VISIT AGAIN !!!

3.2.2 Creation of text file and write sell invoice to the file

Name	Date modified	Type	Size
__pycache__	7/27/2023 6:31 PM	File folder	
Alex Gustavo Costa_sell_invoice	7/27/2023 9:56 PM	Text Document	1 KB
laptop info	7/27/2023 9:53 PM	Text Document	1 KB
main	7/27/2023 6:56 AM	Python Source File	11 KB
operation	7/27/2023 6:56 AM	Python Source File	11 KB
read	7/26/2023 2:32 PM	Python Source File	12 KB
Toshiba Laptop Store_order_invoice	7/27/2023 8:50 PM	Text Document	2 KB
write	7/27/2023 6:56 AM	Python Source File	13 KB

```

Alex Gustavo Costa_sell_invoice
File Edit View
=====
KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603
=====
SELL INVOICE
=====
CUSTOMER NAME: Alex Gustavo Costa
PHONE NUMBER: 9988234456
Date: 27/7/2023 21:52:11
=====
LAPTOP BRAND  LAPTOP MODEL  QUANTITY  UNIT PRICE  AMOUNT
-----
Alienware      Alienware      6          $1978      $11868
Apple          Macbook Pro 16 20  $3500      $70000
-----
GROSS AMOUNT: $82068
SHIPPING COST: $200
-----
TOTAL: $82268
=====
THANK YOU FOR SHOPPING WITH US !!!
VISIT AGAIN !!!
=====
Ln 30, Col 117  100%  Windows (CRLF)  UTF-8

```

3.2.3 Termination of sell process

```
KAPAN, KATHMANDU | CONTACT: 01-4270603

=====
                        SELL INVOICE
=====
CUSTOMER NAME: Alex Gustavo Costa
PHONE NUMBER: 9988234456
Date: 27/7/2023  21:52:11

=====

LAPTOP BRAND  LAPTOP MODEL  QUANTITY  UNIT PRICE  AMOUNT
-----
Alienware     Alienware    6         $1978      $11868
Apple        Macbook Pro 16 20    $3500     $70000
-----
GROSS AMOUNT: $82068
SHIPPING COST: $200
-----
TOTAL: $82268
=====

THANK YOU FOR SHOPPING WITH US !!!
VISIT AGAIN !!!

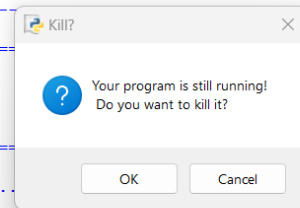
=====

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM

=====
ENTER YOUR SELECTION TO CONTINUE OPERATION: 4
=====

THE SYSTEM HAS BEEN TERMINATED.
=====
```



4.0 Testing

4.1 Testing 1 : To check implementation and functionality of try, except

Test No	1
Objective	To check implementation and functionality of try, except
Action	<ul style="list-style-type: none"> ✓ The main.py module was executed and it displayed the shop title, address and options to select. ✓ The string value was entered in option input prompt instead of integer.
Expected Result	<p>An error message</p> <p>ERROR !!! ENTERED VALUE IS NOT NUMERIC. PLEASE ENTER A NUMRIC VALUE.</p> <p>should display on the screen.</p>
Actual Result	<p>An error message</p> <p>ERROR !!! ENTERED VALUE IS NOT NUMERIC. PLEASE ENTER A NUMRIC VALUE.</p> <p>was displayed on the screen.</p>
Conclusion	Test Successful.

Table 1 Testing 1 : To check implementation and functionality of try, except

```

"""
Using try except block to check if the entered value is integer or not.
If the value is not integer an error message is displayed.
"""
try:
    return int(input_from_user)
except ValueError:
    print(".....")
    print("")
    print("      ERROR !!!")
    print("      ENTERED VALUE IS NOT NUMERIC.")
    print("      PLEASE ENTER A NUMRIC VALUE.")
    print("")
    print(".....")

```

Figure 7 Test 1 : Use of try except block to validate input

```
.....
                                KARKI ELECTRONICS AND LAPTOP SUPPLIERS
                                KAPAN, KATHMANDU | CONTACT: 01-4270603
                                .....

                                WELCOME TO KARKI ELECTRONICS AND LAPTOP SUPPLIERS...
                                .....

                                PLEASE ENTER THE OPTIONS TO CONTINUE...

                                PRESS 1 : TO ORDER LAPTOP
                                PRESS 2 : TO SELL LAPTOP
                                PRESS 3 : TO CHECK LAPTOP STOCK
                                PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM
                                .....

                                ENTER YOUR SELECTION TO CONTINUE OPERATION: hello
                                .....

                                ERROR !!!
                                ENTERED VALUE IS NOT NUMERIC.
                                PLEASE ENTER A NUMRIC VALUE.
                                .....

                                ENTER YOUR SELECTION TO CONTINUE OPERATION:
```

Figure 8 Test 2 : Displaying error message for wrong value

4.2 Testing 2 : To select Order and Sell option for the laptop

4.2.1 Testing 2.1 : To input non existing value in order laptop

Test No	2.1
Objective	To input non existing value in order laptop
Action	<ul style="list-style-type: none"> ✓ The main.py module was executed and it displayed the shop title, address and options to select. ✓ Option 1 was entered to enable order operation process. ✓ An invalid index value "6" was entered.
Expected Result	<p>An error message</p> <p style="text-align: center;">ERROR !!! INVALID LAPTOP INDEX.</p> <p>should display on the screen.</p>
Actual Result	<p>An error message</p> <p style="text-align: center;">ERROR !!! INVALID LAPTOP INDEX.</p> <p>was displayed on the screen.</p>
Conclusion	Test Successful.

Table 2 Testing 2.1 : To input non existing value in order laptop

```

.....
KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603
.....

WELCOME TO KARKI ELECTRONICS AND LAPTOP SUPPLIERS...

.....

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM

.....

ENTER YOUR SELECTION TO CONTINUE OPERATION: 1

.....

WHICH LAPTOP DO YOU WANT TO ORDER ?

.....

WHICH LAPTOP DO YOU WANT TO ORDER ?

.....

+-----+-----+-----+-----+-----+-----+-----+
| S.N | MODEL | BRAND | PRICE | QUANTITY | PROCESSOR | GRAPHICS |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Razer Blade | Razer | $2000.0 | 448 | i7 7th Gen | GTX 3060 |
| 2 | XPS | Dell | $1976.0 | 951 | i5 9th Gen | GTX 3070 |
| 3 | Alienware | Alienware | $1978.0 | 224 | i5 9th Gen | GTX 3070 |
| 4 | Swift 7 | Acer | $900.0 | 650 | i5 9th Gen | GTX 3070 |
| 5 | Macbook Pro 16 | Apple | $3500.0 | 480 | i5 9th Gen | GTX 3070 |
+-----+-----+-----+-----+-----+-----+-----+

.....

ENTER THE INDEX OF LAPTOP: 6

.....

ERROR !!!
INVALID LAPTOP INDEX.

.....

ENTER THE INDEX OF LAPTOP:

```

Figure 9 Testing 2.1 : Displaying error message for index which is not available during laptop order process

4.2.2 Testing 2.2 : To input non existing value in sell laptop

Test No	2.2
Objective	To input non existing value in sell laptop
Action	<ul style="list-style-type: none"> ✓ The main.py module was executed and it displayed the shop title, address and options to select. ✓ Option 2 was entered to enable sell operation process. ✓ An invalid index value “10” was entered.
Expected Result	<p>An error message</p> <p style="text-align: center;">ERROR !!! INVALID LAPTOP INDEX.</p> <p>should display on the screen.</p>
Actual Result	<p>An error message</p> <p style="text-align: center;">ERROR !!! INVALID LAPTOP INDEX.</p> <p>was displayed on the screen.</p>
Conclusion	Test Successful.

Table 3 Testing 2.2 : To input non existing value in sell laptop

```

.....
KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603
.....

WELCOME TO KARKI ELECTRONICS AND LAPTOP SUPPLIERS...
.....

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM
.....

ENTER YOUR SELECTION TO CONTINUE OPERATION: 2
.....

WHICH LAPTOP DO YOU WANT TO SELL?
.....

+-----+-----+-----+-----+-----+-----+-----+
| S.N | MODEL | BRAND | PRICE | QUANTITY | PROCESSOR | GRAPHICS |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Razer Blade | Razer | $2000.0 | 446 | i7 7th Gen | GTX 3060 |
| 2 | XPS | Dell | $1976.0 | 951 | i5 9th Gen | GTX 3070 |
| 3 | Alienware | Alienware | $1978.0 | 224 | i5 9th Gen | GTX 3070 |
| 4 | Swift 7 | Acer | $900.0 | 650 | i5 9th Gen | GTX 3070 |
| 5 | Macbook Pro 16 | Apple | $3500.0 | 480 | i5 9th Gen | GTX 3070 |
+-----+-----+-----+-----+-----+-----+-----+

.....

ENTER THE INDEX OF LAPTOP: 10
.....

ERROR !!!
INVALID LAPTOP INDEX.
.....

ENTER THE INDEX OF LAPTOP:

```

Figure 10 Testing 2.2 : Displaying error message for index which is not available during laptop sell process

4.2.2 Testing 2.3 : To input negative value in order laptop

Test No	2.3
Objective	To input non existing value in sell laptop
Action	<ul style="list-style-type: none"> ✓ The main.py module was executed and it displayed the shop title, address and options to select. ✓ Option 1 was entered to enable order operation process. ✓ An invalid index value “-5” was entered.
Expected Result	<p>An error message</p> <p style="text-align: center;">ERROR !!! INVALID LAPTOP INDEX.</p> <p>should display on the screen.</p>
Actual Result	<p>An error message</p> <p style="text-align: center;">ERROR !!! INVALID LAPTOP INDEX.</p> <p>was displayed on the screen.</p>
Conclusion	Test Successful.

Table 4 Testing 2.3 : To input negative value in order laptop

```

.....
KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603
.....

WELCOME TO KARKI ELECTRONICS AND LAPTOP SUPPLIERS...
.....

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM
.....

ENTER YOUR SELECTION TO CONTINUE OPERATION: 1
.....

WHICH LAPTOP DO YOU WANT TO ORDER ?
.....

+-----+-----+-----+-----+-----+-----+-----+
| S.N | MODEL | BRAND | PRICE | QUANTITY | PROCESSOR | GRAPHICS |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Razer Blade | Razer | $2000.0 | 446 | i7 7th Gen | GTX 3060 |
| 2 | XPS | Dell | $1976.0 | 951 | i5 9th Gen | GTX 3070 |
| 3 | Alienware | Alienware | $1978.0 | 224 | i5 9th Gen | GTX 3070 |
| 4 | Swift 7 | Acer | $900.0 | 650 | i5 9th Gen | GTX 3070 |
| 5 | Macbook Pro 16 | Apple | $3500.0 | 480 | i5 9th Gen | GTX 3070 |
+-----+-----+-----+-----+-----+-----+-----+

ENTER THE INDEX OF LAPTOP: -5
.....

ERROR !!!
INVALID INDEX
PLEASE ENTER A VALID NUMERIC LAPTOP INDEX.
.....

ENTER THE INDEX OF LAPTOP:

```

Figure 11 Testing 2.3 : Displaying error message if index is negative for order process

4.2.2 Testing 2.4 : To input negative value in sell laptop

Test No	2.4
Objective	To input non existing value in sell laptop
Action	<ul style="list-style-type: none"> ✓ The main.py module was executed and it displayed the shop title, address and options to select. ✓ Option 2 was entered to enable sell operation process. ✓ An invalid index value “-1” was entered.
Expected Result	<p>An error message</p> <p style="text-align: center;">ERROR !!! INVALID LAPTOP INDEX.</p> <p>should display on the screen.</p>
Actual Result	<p>An error message</p> <p style="text-align: center;">ERROR !!! INVALID LAPTOP INDEX.</p> <p>was displayed on the screen.</p>
Conclusion	Test Successful.

Table 5 4.2.2 Testing 2.4 : To input negative value in sell laptop

```

.....
KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603
.....

WELCOME TO KARKI ELECTRONICS AND LAPTOP SUPPLIERS...

.....

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM

.....

ENTER YOUR SELECTION TO CONTINUE OPERATION: 2
.....

WHICH LAPTOP DO YOU WANT TO SELL?

.....
+-----+-----+-----+-----+-----+-----+-----+
| S.N | MODEL | BRAND | PRICE | QUANTITY | PROCESSOR | GRAPHICS |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Razer Blade | Razer | $2000.0 | 446 | i7 7th Gen | GTX 3060 |
| 2 | XPS | Dell | $1976.0 | 951 | i5 9th Gen | GTX 3070 |
| 3 | Alienware | Alienware | $1978.0 | 224 | i5 9th Gen | GTX 3070 |
| 4 | Swift 7 | Acer | $900.0 | 650 | i5 9th Gen | GTX 3070 |
| 5 | Macbook Pro 16 | Apple | $3500.0 | 480 | i5 9th Gen | GTX 3070 |
+-----+-----+-----+-----+-----+-----+-----+

.....

ENTER THE INDEX OF LAPTOP: -1
.....

ERROR !!!
INVALID INDEX
PLEASE ENTER A VALID NUMERIC LAPTOP INDEX.

.....

ENTER THE INDEX OF LAPTOP:

```

Figure 12 Testing 2.4 : Displaying error message if index is negative for sell process

4.3 Testing 3 : To display file generation of order laptops(Order multiple laptops)

Test No	3
Objective	To display file generation of order laptops(Sell multiple laptops)
Action	<ul style="list-style-type: none"> ✓ The main.py module was executed and it displayed the shop title, address and options to select. ✓ Option 1 was entered to enable order operation process and display the table ✓ Index and quantity of laptop you want to sell and display total amount was asked to display ✓ Generate an invoice with following details <p>Distributor Name : Alex Gustavo Costa Phone number : 9845342180 Laptop Brand : Razer Laptop Model : Razer Blade Laptop Quantity : 3 Laptop Price : \$2000</p> <p>Add more laptops</p> <p>Laptop Brand : Dell Laptop Model : XPS Laptop Quantity : 5 Laptop Price : \$1976</p> <p>Don't add more laptop</p> ✓ Generate order invoice and write order invoice to text file
Expected Result	<ul style="list-style-type: none"> ✓ The main.py module should executed and it displayed the shop title, address and options to select. ✓ Option 1 should be entered to enable order operation process and display the table ✓ Index and quantity of laptop you want to sell and display total amount should be taken and displayed ✓ An invoice with following details should be generated

	<p>Distributor Name : Alex Gustavo Costa Phone number : 9845342180 Laptop Brand : Razer Laptop Model : Razer Blade Laptop Quantity : 3 Laptop Price : \$2000</p> <p>More laptops should be added</p> <p>Laptop Brand : Dell Laptop Model : XPS Laptop Quantity : 5 Laptop Price : \$1976</p> <p>More laptop should not be added.</p> <p>✓ An order invoice should be generated and written to order invoice to text file</p>
Actual Result	Every task in Expected result was conducted
Conclusion	Test Successful.

Table 6 Testing 3 : To display file generation of order laptops(Order multiple laptops)

```

.....

KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603

.....

WELCOME TO KARKI ELECTRONICS AND LAPTOP SUPPLIERS...

.....

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM

.....

ENTER YOUR SELECTION TO CONTINUE OPERATION: 1

```


WHICH LAPTOP DO YOU WANT TO ORDER ?

S.N	MODEL	BRAND	PRICE	QUANTITY	PROCESSOR	GRAPHICS
1	Razer Blade	Razer	\$2000.0	451	i7 7th Gen	GTX 3060
2	XPS	Dell	\$1976.0	956	i5 9th Gen	GTX 3070
3	Alienware	Alienware	\$1978.0	230	i5 9th Gen	GTX 3070
4	Swift 7	Acer	\$900.0	650	i5 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	\$3500.0	500	i5 9th Gen	GTX 3070

ENTER THE INDEX OF LAPTOP: 1
ENTER QUANTITY:3

LAPTOP QUNATITY UPDATED IN THE TEXT FILE.

TOTAL AMOUNT : \$6000.0

DO YOU WANT TO BUY MORE LAPTOPS? (YES/NO): yes

ENTER THE INDEX OF LAPTOP: 2
ENTER QUANTITY:5

LAPTOP QUNATITY UPDATED IN THE TEXT FILE.

TOTAL AMOUNT : \$9880.0

DO YOU WANT TO BUY MORE LAPTOPS? (YES/NO): no

THANK YOU FOR SHOPPING WITH US !
VISIT AGAIN !!!

TO GENERATE ORDER INVOICE PLEASE ENTER THE FOLLOWING DETAILS

ENTER DISTRIBUTOR NAME: Toshiba Laptop Store

```

ENTER PHONE NUMBER: 9845342180
ENTER LAPTOP BRAND: Razer
ENTER LAPTOP MODEL: Razer Blade
ENTER LAPTOP QUANTITY: 3
ENTER LAPTOP PRICE: $2000
DO YOU WANT TO ORDER MORE LAPTOPS? (YES/NO): yes
ENTER LAPTOP BRAND: Dell
ENTER LAPTOP MODEL: XPS
ENTER LAPTOP QUANTITY: 5
ENTER LAPTOP PRICE: $1976
DO YOU WANT TO ORDER MORE LAPTOPS? (YES/NO): no

```

```

=====
                                KARKI ELECTRONICS AND LAPTOP SUPPLIERS
                                KAPAN, KATHMANDU | CONTACT: 01-4270603
=====
                                ORDER INVOICE
=====
Name of customer: Toshiba Laptop Store

Phone no: 9845342180

Date: 27/7/2023  20:13:16

=====
LAPTOP BRAND  LAPTOP MODEL  QUANTITY  UNIT PRICE  AMOUNT
-----
Razer         Razer Blade  3         $2000      $6000
Dell          XPS          5         $1976      $9880
-----
                                Gross Amount: $15880

                                13% VAT: $2064.4

                                Total Amount: $17944.4

=====
                                THANK YOU FOR SHOPPING WITH US !!!
                                VISIT AGAIN !!!
=====

```

Figure 13 Testing 3 : Conducting order process and displaying invoice on screen

Name	Date modified	Type	Size
__pycache__	7/27/2023 6:31 PM	File folder	
laptop info	7/27/2023 8:14 PM	Text Document	1 KB
main	7/27/2023 6:56 AM	Python Source File	11 KB
operation	7/27/2023 6:56 AM	Python Source File	11 KB
read	7/26/2023 2:32 PM	Python Source File	12 KB
Toshiba Laptop Store_order_invoice	7/27/2023 8:16 PM	Text Document	2 KB
write	7/27/2023 6:56 AM	Python Source File	13 KB

Figure 14 Testing 3 : Allocation of text file after order invoice is generated and written to text file

Discussion and Analysis

Toshiba Laptop Store_order_invoice

+

File

Edit

View

=====

KARKI ELECTRONICS AND LAPTOP SUPPLIERS

KAPAN, KATHMANDU | CONTACT: 01-4270603

=====

ORDER INVOICE

=====

Name of customer: Toshiba Laptop Store

Phone no: 9845342180

Date: 27/7/2023 20:13:16

=====

LAPTOP BRAND	LAPTOP MODEL	QUANTITY	UNIT PRICE	AMOUNT
Razer	Razer Blade	3	\$2000	\$6000
Dell	XPS	5	\$1976	\$9880

Gross Amount:	\$15880
13% VAT:	\$2064.4
Total Amount:	\$17944.4

=====

THANK YOU FOR SHOPPING WITH US !!!

VISIT AGAIN !!!

=====

Ln 7, Col 254

100%

Windows (CRLF)

UTF-8

Figure 15 Testing 3 : Written order invoice in text file

4.4 Testing 4 : To display file generation of sell laptops(Sell multiple laptops)

Test No	4
Objective	To display file generation of sell laptops(Sell multiple laptops)
Action	<ul style="list-style-type: none"> ✓ The main.py module was executed and it displayed the shop title, address and options to select. ✓ Option 2 was entered to enable sell operation process and display the table ✓ Index and quantity of laptop you want to sell and display total amount was asked to display ✓ Generate an invoice with following details <p>Customer Name : Alex Gustavo Costa Phone number : 9988234456 Laptop Brand : Alienware Laptop Model : Alienware Laptop Quantity : 6 Laptop Price : \$1978</p> <p>Add more laptops</p> <p>Laptop Brand : Apple Laptop Model : Macbook Pro 16 Laptop Quantity : 20 Laptop Price : \$3500</p> <p>Don't add more laptop Ask shipping and add \$200</p> ✓ Generate sell invoice and write sell invoice to text file
Expected Result	<ul style="list-style-type: none"> ✓ The main.py module should executed and it displayed the shop title, address and options to select. ✓ Option 2 should be entered to enable sell operation process and display the table ✓ Enter Index and quantity of laptop you want to sell and display total amount ✓ Generate an invoice with following details

	<p>Customer Name : Alex Gustavo Costa Phone number : 9988234456 Laptop Brand : Alienware Laptop Model : Alienware Laptop Quantity : 6 Laptop Price : \$1978</p> <p>More laptops should be added</p> <p>Laptop Brand : Apple Laptop Model : Macbook Pro 16 Laptop Quantity : 20 Laptop Price : \$3500</p> <p>More laptop should not be added Ask shipping and add \$200</p> <ul style="list-style-type: none"> ✓ Shipping should be added to the total amount ✓ Sell invoice should be generated and written to sell invoice text file
Actual Result	Every task in Expected result was conducted
Conclusion	Test Successful.

Table 7 Testing 4 : To display file generation of sell laptops(Sell multiple laptops)

```

.....
KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603
.....

WELCOME TO KARKI ELECTRONICS AND LAPTOP SUPPLIERS...
.....
.....

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM
.....
.....

ENTER YOUR SELECTION TO CONTINUE OPERATION: 2
.....
.....

WHICH LAPTOP DO YOU WANT TO SELL?
.....
.....
+-----+-----+-----+-----+-----+-----+-----+
| S.N | MODEL | BRAND | PRICE | QUANTITY | PROCESSOR | GRAPHICS |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Razer Blade | Razer | $2000.0 | 448 | i7 7th Gen | GTX 3060 |
| 2 | XPS | Dell | $1976.0 | 951 | i5 9th Gen | GTX 3070 |
| 3 | Alienware | Alienware | $1978.0 | 230 | i5 9th Gen | GTX 3070 |
| 4 | Swift 7 | Acer | $900.0 | 650 | i5 9th Gen | GTX 3070 |
| 5 | Macbook Pro 16 | Apple | $3500.0 | 500 | i5 9th Gen | GTX 3070 |
+-----+-----+-----+-----+-----+-----+-----+
.....
ENTER THE INDEX OF LAPTOP: 3
ENTER QUANTITY:6
.....
.....

LAPTOP QUNATITY UPDATED IN THE TEXT FILE.
.....
.....

TOTAL AMOUNT : $11868.0
.....
DO YOU WANT TO BUY MORE LAPTOPS? (YES/NO): yes
.....
ENTER THE INDEX OF LAPTOP: 5
ENTER QUANTITY:20
.....
.....

LAPTOP QUNATITY UPDATED IN THE TEXT FILE.
.....
.....

TOTAL AMOUNT : $70000.0
.....
DO YOU WANT TO BUY MORE LAPTOPS? (YES/NO): no
.....
.....

```

```

THANK YOU FOR SHOPPING WITH US !
VISIT AGAIN !!!

.....

=====
TO GENERATE ORDER INVOICE PLEASE ENTER THE FOLLOWING DETAILS
=====

ENTER CUSTOMER NAME: Alex Gustavo Costa
ENTER PHONE NUMBER: 9988234456
ENTER LAPTOP BRAND: Alienware
ENTER LAPTOP MODEL: Alienware
ENTER LAPTOP QUANTITY: 6
ENTER LAPTOP PRICE: $1978
DO YOU WANT TO PURCHASE MORE LAPTOPS? (YES/NO): yes
ENTER LAPTOP BRAND: Apple
ENTER LAPTOP MODEL: Macbook Pro 16
ENTER LAPTOP QUANTITY: 20
ENTER LAPTOP PRICE: $3500
DO YOU WANT TO PURCHASE MORE LAPTOPS? (YES/NO): no
DO YOU WANT SHIPPING (YES/NO): yes

=====

KARKI ELECTRONICS AND LAPTOP SUPPLIERS
KAPAN, KATHMANDU | CONTACT: 01-4270603

=====

SELL INVOICE

=====
CUSTOMER NAME: Alex Gustavo Costa
PHONE NUMBER: 9988234456
Date: 27/7/2023 21:52:11

=====

LAPTOP BRAND  LAPTOP MODEL  QUANTITY  UNIT PRICE  AMOUNT
-----
Alienware      Alienware      6          $1978      $11868
Apple          Macbook Pro 16 20      $3500      $70000
-----
GROSS AMOUNT:  $82068
SHIPPING COST:  $200
-----
TOTAL: $82268
=====

THANK YOU FOR SHOPPING WITH US !!!
VISIT AGAIN !!!

=====

```

Figure 16 Testing 4 : Conducting sell process and displaying sell invoice on the screen

Name	Date modified	Type	Size
__pycache__	7/27/2023 6:31 PM	File folder	
Alex Gustavo Costa_sell_invoice	7/27/2023 9:56 PM	Text Document	1 KB
laptop info	7/27/2023 9:53 PM	Text Document	1 KB
main	7/27/2023 6:56 AM	Python Source File	11 KB
operation	7/27/2023 6:56 AM	Python Source File	11 KB
read	7/26/2023 2:32 PM	Python Source File	12 KB
Toshiba Laptop Store_order_invoice	7/27/2023 8:50 PM	Text Document	2 KB
write	7/27/2023 6:56 AM	Python Source File	13 KB

Figure 17 Testing 4 : Allocating sell invoice text file after generating and displaying sell invoice on screen

KARKI ELECTRONICS AND LAPTOP SUPPLIERS KAPAN, KATHMANDU CONTACT: 01-4270603				
SELL INVOICE				
CUSTOMER NAME: Alex Gustavo Costa PHONE NUMBER: 9988234456 Date: 27/7/2023 21:52:11				
LAPTOP BRAND	LAPTOP MODEL	QUANTITY	UNIT PRICE	AMOUNT
Alienware	Alienware	6	\$1978	\$11868
Apple	Macbook Pro 16 20		\$3500	\$70000
GROSS AMOUNT: \$82068				
SHIPPING COST: \$200				
TOTAL: \$82268				
THANK YOU FOR SHOPPING WITH US !!! VISIT AGAIN !!!				

Figure 18 Testing 4 : Written sell invoice to a text file

4.5 Testing 5 : To update the quantity being deducted in a text file while ordering or selling laptop

Test No	5
Objective	To update the quantity being deducted in a text file while ordering or selling laptop
Action	<ul style="list-style-type: none"> ✓ The main.py module was executed and it displayed the shop title, address and options to select. ✓ Option 1 was entered to enable order operation process. ✓ Laptop index was entered as 2 ✓ Laptop quantity was entered as 5 ✓ Option 2 was entered to enable sell operation process. ✓ Laptop index was entered as 3 ✓ Laptop quantity was entered as 9
Expected Result	<ul style="list-style-type: none"> ✓ 5 Laptops should be deducted from index 2 in order operation. ✓ 9 Laptops should be deducted from index 3 in sell operation.
Actual Result	<ul style="list-style-type: none"> ✓ 5 Laptops were deducted from index 2 in order operation. ✓ 9 Laptops were deducted from index 3 in sell operation.
Conclusion	Test Successful.

Table 8 4.5 Testing 5 : To update the quantity being deducted in a text file while ordering or selling laptop

```

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM

.....
ENTER YOUR SELECTION TO CONTINUE OPERATION: 2
.....

WHICH LAPTOP DO YOU WANT TO 2 ORDER ?
.....

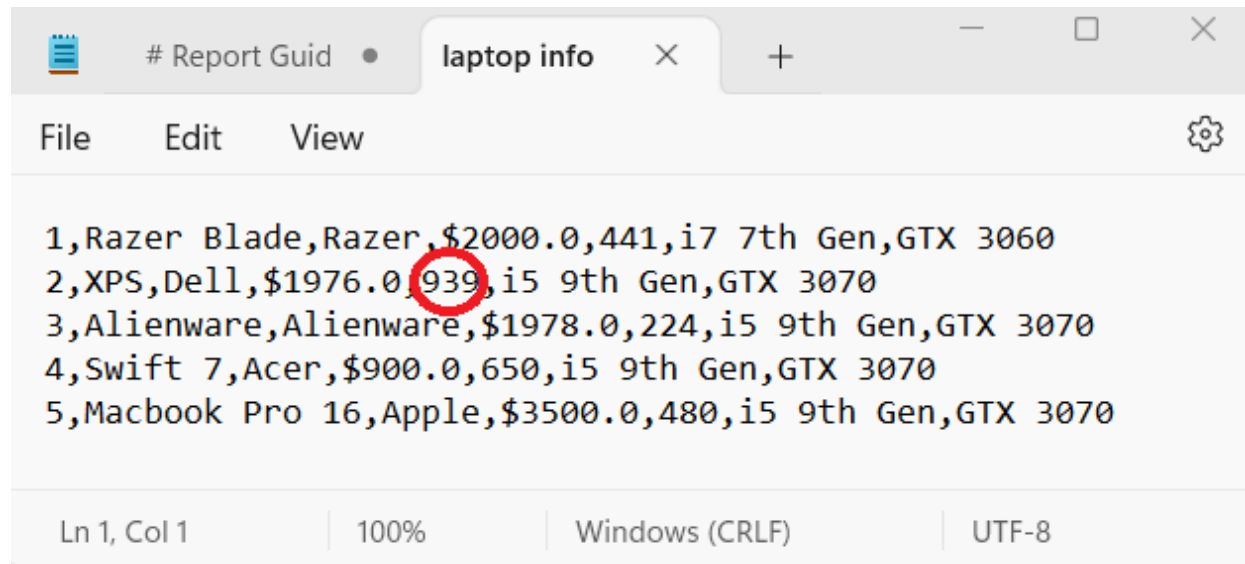
+-----+-----+-----+-----+-----+-----+-----+
| S.N | MODEL | BRAND | PRICE | QUANTITY | PROCESSOR | GRAPHICS |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Razer Blade | Razer | $2000.0 | 441 | i7 7th Gen | GTX 3060 |
| 2 | XPS | Dell | $1976.0 | 944 | i5 9th Gen | GTX 3070 |
| 3 | Alienware | Alienware | $1978.0 | 224 | i5 9th Gen | GTX 3070 |
| 4 | Swift 7 | Acer | $900.0 | 650 | i5 9th Gen | GTX 3070 |
| 5 | Macbook Pro 16 | Apple | $3500.0 | 480 | i5 9th Gen | GTX 3070 |
+-----+-----+-----+-----+-----+-----+-----+

ENTER THE INDEX OF LAPTOP: 2
ENTER QUANTITY:5
.....

LAPTOP QUNATITY UPDATED IN THE TEXT FILE.

```

Figure 19 Testing 5 : Displaying Laptop quantity update on screen for order operation



The screenshot shows a text editor window titled "laptop info" with a menu bar (File, Edit, View) and a status bar (Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8). The text content is a list of five laptops with their details separated by commas. The quantity for the second laptop, "XPS", has been updated from 944 to 939, which is circled in red.

```

1,Razer Blade,Razer,$2000.0,441,i7 7th Gen,GTX 3060
2,XPS,Dell,$1976.0,939,i5 9th Gen,GTX 3070
3,Alienware,Alienware,$1978.0,224,i5 9th Gen,GTX 3070
4,Swift 7,Acer,$900.0,650,i5 9th Gen,GTX 3070
5,Macbook Pro 16,Apple,$3500.0,480,i5 9th Gen,GTX 3070

```

Figure 20 Testing 5 : Displaying Laptop quantity update on text file for order operation

```

PLEASE ENTER THE OPTIONS TO CONTINUE...

PRESS 1 : TO ORDER LAPTOP
PRESS 2 : TO SELL LAPTOP
PRESS 3 : TO CHECK LAPTOP STOCK
PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE SYSTEM

.....
ENTER YOUR SELECTION TO CONTINUE OPERATION: 2
.....

WHICH LAPTOP DO YOU WANT TO SELL?

.....
+-----+-----+-----+-----+-----+-----+-----+
| S.N | MODEL | BRAND | PRICE | QUANTITY | PROCESSOR | GRAPHICS |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Razer Blade | Razer | $2000.0 | 441 | i7 7th Gen | GTX 3060 |
| 2 | XPS | Dell | $1976.0 | 939 | i5 9th Gen | GTX 3070 |
| 3 | Alienware | Alienware | $1978.0 | 224 | i5 9th Gen | GTX 3070 |
| 4 | Swift 7 | Acer | $900.0 | 650 | i5 9th Gen | GTX 3070 |
| 5 | Macbook Pro 16 | Apple | $3500.0 | 480 | i5 9th Gen | GTX 3070 |
+-----+-----+-----+-----+-----+-----+-----+

ENTER THE INDEX OF LAPTOP: 3
ENTER QUANTITY:9

.....
LAPTOP QUNATITY UPDATED IN THE TEXT FILE.

```

Figure 21 Testing 5 : Displaying Laptop quantity update on screen for sell operation

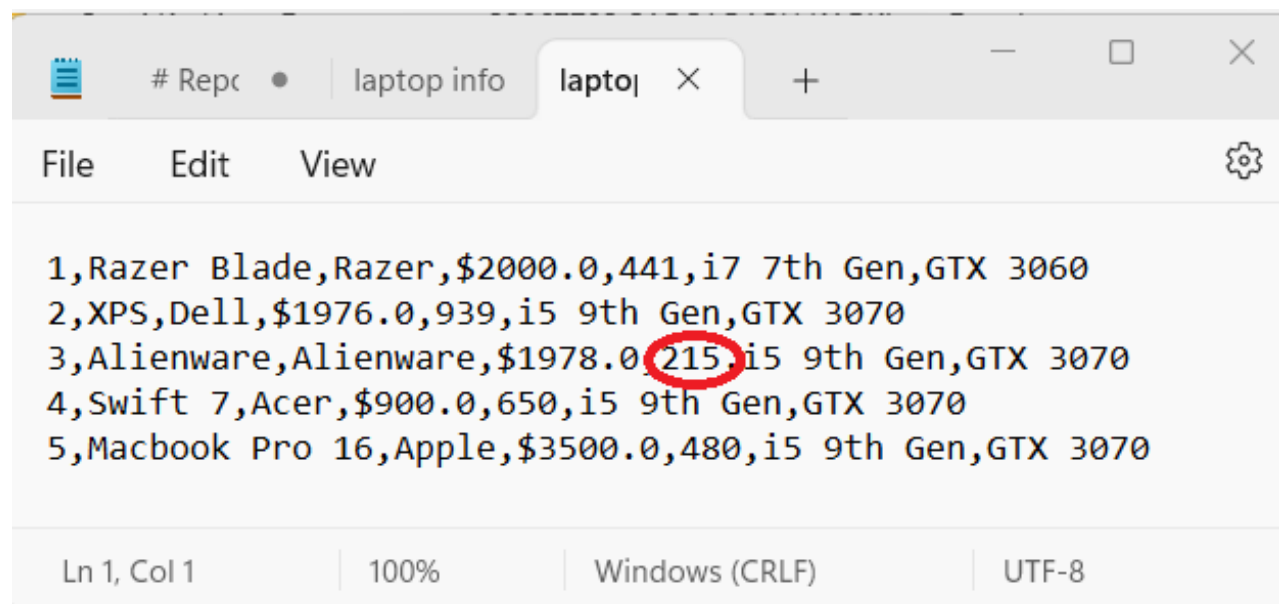


Figure 22 Testing 5 : Displaying Laptop quantity update on text file for sell operation

5.0 Conclusion

From this project we are able to learn about different components used to develop a billing system for a laptop shop. We are able to develop the system with the help of python programming language. This project have helped me gain knowledge about effective tracking of laptop rental and buying, invoice generation and manage the records of the customers.

During the process of development, I had encountered many errors and bugs in my coding project. At that time some valuable sources were useful to me. At the same time my friends, teachers and my close ones helped me in developing laptop management system.

Utilizing the Python programming language, we were able to develop a billing system, which is user-friendly and meets the requirement of the business goals. This system is designed and developed to fulfil the requirement of the laptop store. Additionally, using python tools like dictionary, file read and write operations, loops and other necessary tools, we were able to develop a functional system.

Overall, this laptop shop billing system development project was proven to be successful and improved the efficiency, accuracy and productivity of the laptop store. The project also helped in demonstrating the importance of python programming in software development, mostly for business applications.

The project also helped us to be knowledgeable about the versatility of python programming language as well.

6.0 References

Anon., 2023. *Asana*. [Online]

Available at: <https://asana.com/resources/what-is-a-flowchart>
[Accessed 25 7 2023].

Anon., 2023. *edureka*. [Online]

Available at: <https://www.edureka.co/blog/data-structures-in-python/>
[Accessed 26 07 2023].

Anon., 2023. *edureka*. [Online]

Available at: <https://www.edureka.co/blog/data-structures-in-python/#tuple>
[Accessed 26 07 2023].

Anon., 2023. *edureka*. [Online]

Available at: <https://www.edureka.co/blog/data-structures-in-python/#dictionary>
[Accessed 26 07 2023].

Anon., 2023. *edureka*. [Online]

Available at: <https://www.edureka.co/blog/data-structures-in-python/#set>
[Accessed 26 07 2023].

Anon., 2023. *Geek for Geeks*. [Online]

Available at: <https://www.geeksforgeeks.org/python-programming-language/>
[Accessed 10 05 2023].

Anon., 2023. *GeeksforGeeks*. [Online]

Available at: <https://www.geeksforgeeks.org/python-data-structures/>
[Accessed 25 07 2023].

S.Gillis, A., 2022. *Tech Target*. [Online]

Available at: <https://www.techtarget.com/whatis/definition/algorithm>
[Accessed 10 05 2023].

6.0 Appendix

6.0.1 Appendix : Code for main.py module

#Import all the required libraries to this main module

import read

import write

import operation

def main_screen_display():

 # Display the title, address and contact details of the shop and greet the user.

 print(".....")

 print("")

 print(" KARKI ELECTRONICS AND LAPTOP SUPPLIERS")

 print(" KAPAN, KATHMANDU | CONTACT: 01-4270603")

 print("")

 print(".....")

 print("")

 print(" WELCOME TO KARKI ELECTRONICS AND LAPTOP
SUPPLIERS...")

 print("")

 print(".....")

def validate_input_from_user():

 while True:

 """

 To conduct order and selling process a
 screen with options are displayed for

selecting the operation the user want to
continue and the user have to enter the value.

If the user enter the value apart from the displayed screen,
an error message is displayed.

```

"""
print(".....")
input_from_user = input("  ENTER YOUR SELECTION TO CONTINUE
OPERATION: ")
print(".....")

"""

Check whether the field is empty or not
otherwise display suitable message and ask
the user to input the value again.
"""

if not input_from_user.strip():
    print(".....")
    print("")
    print("  ERROR !!!")
    print("  INVALID INPUT.")
    print("  PLEASE ENTER VALUE FROM 1 TO 4.")
    print("")
    print(".....")
    continue

"""

```

Using try except block to check if the entered value is integer or not.
If the value is not integer an error message is displayed.

```

"""

try:
    return int(input_from_user)

except ValueError:
    print(".....")
    print("")
    print("    ERROR !!!")
    print("    ENTERED VALUE IS NOT NUMERIC.")
    print("    PLEASE ENTER A NUMRIC VALUE.")
    print("")
    print(".....")

if __name__ == "__main__":
    while True:
        main_screen_display()

    loop_selection_user = True

    while loop_selection_user:
        print(".....")
        print("")
        print("    PLEASE ENTER THE OPTIONS TO CONTINUE...")
        print("")
        print("    PRESS 1 : TO ORDER LAPTOP")
        print("    PRESS 2 : TO SELL LAPTOP")
        print("    PRESS 3 : TO CHECK LAPTOP STOCK")

```



```

    print("  PRESS 4 : TO CLOSE THE PROGRAM AND EXIT FROM THE
SYSTEM")
    print("")
    print(".....")

    option_selection = validate_input_from_user()

    if option_selection > 4 or option_selection < 1:

        print(".....")
        print("")
        print("  ERROR !!! ")
        print("  INVALID SELECTION.")
        print("  PLEASE SELECT FROM 1 TO 4.")
        print("")

        print(".....")

    elif option_selection == 1:
        """
        When the user enters 1 as the input a question is displayed
        asking which laptop the user want to order.
        Along with the question the list of laptop is displayed in
        tabular format.

        Functions from other modules are called to this file to
        display the list of laptops, take input from the user and
        display the output in the form of invoice after ordering the laptop.
        """

```

```
print(".....")

    print("")
    print("  WHICH LAPTOP DO YOU WANT TO ORDER ?")
    print("")

print(".....")

    read.display_laptop_list_table()
    read.laptop_quantity_update()

try:
    # Attempt to get order calculation details
    distributor_name, laptop_list, distributor_laptop_gross_amount,
distributor_total_invoice_amount, distributor_phone_no =
operation.get_order_calculation()

    # Check if the operation was successful (no validation errors)
    if distributor_name is not None and laptop_list is not None:
        # Display the order invoice and write it to a file
        write.generate_and_display_order_invoice(distributor_name, laptop_list,
distributor_laptop_gross_amount, distributor_total_invoice_amount,
distributor_phone_no)

        write.write_order_invoice(distributor_name, laptop_list,
distributor_laptop_gross_amount, distributor_total_invoice_amount,
distributor_phone_no)

except TypeError:
    # Handle the case where operation.get_order_calculation() returned None
(validation error)
    print("")
```

```
elif option_selection == 2:
```

```
    """
```

When the user enters 2 as the input a question is displayed asking which laptop the user want to sell. Along with the question the list of laptop is displayed in tabular format.

Functions from other modules are called to this file to display the list of laptops, take input from the user and display the output in the form of invoice after selling the laptop.

```
    """
```

```
print(".....")
```

```
    print("")
```

```
    print("  WHICH LAPTOP DO YOU WANT TO SELL?")
```

```
    print("")
```

```
print(".....")
```

```
    read.display_laptop_list_table()
```

```
    read.laptop_quantity_update()
```

```
    try:
```

```
        sell_customer_name, laptop_list, sell_total_invoice_amount,
sell_phone_no, sell_shipping_cost = operation.get_sell_calculation()
```

```
        # Check if the operation was successful (no validation errors)
```

```
        if sell_customer_name is not None and laptop_list is not None:
```

```
            # Display the order invoice and write it to a file
```

```

        write.generate_and_display_sell_invoice(sell_customer_name,
laptop_list, sell_total_invoice_amount, sell_phone_no, sell_shipping_cost)

```

```

        write.write_sell_invoice(sell_customer_name, laptop_list,
sell_total_invoice_amount, sell_phone_no, sell_shipping_cost)

```

```

    except TypeError:

```

```

        # Handle the case where operation.get_order_calculation() returned None
(validation error)

```

```

        print("")

```

```

elif option_selection == 3:

```

```

    """

```

```

    When the user enters 3 as the input a message is displayed
    showing the available laptops and its stock available in a
    tabular format along with other laptop details.

```

```

    """

```

```

print(".....")

```

```

    print("")

```

```

    print("  THESE ARE THE LAPTOPS AVAILABLE AT OUR STORE
CURRENTLY...")

```

```

    print("")

```

```

print(".....")

```

```

read.display_laptop_list_table()

```

```

    loop_selection_user = False # Exit the inner loop and go back to the main
menu

```

```

elif option_selection == 4:

```

```
"""
```

When the user enters 4 as the input a message is displayed
saying system have been terminated and stops the operation

```
"""
```

```
print(".....")
```

```
    print("")
```

```
    print(" THE SYSTEM HAS BEEN TERMINATED.")
```

```
    print("")
```

```
print(".....")
```

```
    exit()
```

6.0.2 Appendix : Code for operation.py module

```
import write
```

```
def input_validation(prompt, type_data):
```

```
    """
```

```
    Function is declared to check the validation
    of the values entered by the user.
```

```
    If the user enters the wrong value, an appropriate
    error message is displayed.
```

```
    """
```

```
    while True:
```

```
        input_valid = input(prompt)
```

```
        if not input_valid.strip():
```

```
            # If the input field is empty, display an error message and loop back to the
            prompt.
```

```
            print(".....")
```

```
            print("")
```

```
            print("    ERROR !!!")
```

```
            print("    EMPTY FIELD.")
```

```
            print("    PLEASE ENTER APPROPRIATE VALUE.")
```

```
            print("")
```

```
            print(".....")
```

```
            continue
```

```
        if type_data == str and (input_valid.isdigit() or input_valid.replace(".", "").isdigit()):
```

If the input field has a wrong value entered by the user, display an error message and loop back to the prompt.

```
print(".....")
print("")
print("    ERROR !!!")
print("    INPUT INVALID.")
print("    PLEASE ENTER A STRING VALUE.")
print("")
print(".....")
continue
```

if type_data == int:

If the entered input is not an integer, display an error message and loop back to the prompt.

if not input_valid.isdigit():

```
print(".....")
print("")
print("    ERROR !!!")
print("    INVALID INPUT.")
print("    PLEASE ENTER AN INTEGER VALUE.")
print("")
```

```
print(".....")
continue
```

Convert the input to an integer before returning

return int(input_valid)

if type_data == bool:

If the entered input is not boolean, display an error message and loop back to the prompt.

```
if input_valid.lower() not in ["yes","no"]:
```

```
print(".....")
```

```
    print("")
```

```
    print("    ERROR !!!")
```

```
    print("    INVALID INPUT.")
```

```
    print("    PLEASE ENTER 'YES' OR 'NO'")
```

```
    print("")
```

```
print(".....")
```

```
    continue
```

```
# Convert the input to a boolean before returning
```

```
return input_valid.lower() == 'yes'
```

```
# For any other type (e.g., str), return value as is
```

```
return input_valid
```

```
"""
```

Generation of order invoice by taking all the required details from the distributor and generate the invoice by doing required calculations

```
"""
```

```
def get_order_calculation():
```

```
    distributor_order_result = get_distributor_laptop_order_details()
```

```
# Check if the function returned None due to validation error
```

```
if distributor_order_result is not None:
```



```
distributor_name, laptop_list, distributor_laptop_gross_amount,  
distributor_total_invoice_amount, distributor_phone_no = distributor_order_result
```

```
# Display order invoice on the screen
```

```
write.generate_and_display_order_invoice(distributor_name, laptop_list,  
distributor_laptop_gross_amount, distributor_total_invoice_amount,  
distributor_phone_no)
```

```
# Write the invoice to the text file after data is entered
```

```
write.write_order_invoice(distributor_name, laptop_list,  
distributor_laptop_gross_amount, distributor_total_invoice_amount,  
distributor_phone_no)
```

```
def get_sell_calculation():
```

```
    sell_order_result = get_customer_laptop_sell_details()
```

```
# Check if the function returned None due to validation error
```

```
if sell_order_result is not None:
```

```
    sell_customer_name, laptop_list, sell_total_invoice_amount, sell_phone_no,  
    sell_shipping_cost = sell_order_result
```

```
# Ask if the user wants shipping
```

```
    sell_customer_shipping_choice = input_validation(" DO YOU WANT SHIPPING  
(YES/NO): ", bool)
```

```
if sell_customer_shipping_choice:
```

```
    sell_shipping_cost = 200
```

```
    sell_total_invoice_amount += sell_shipping_cost # Add shipping cost to the total  
invoice amount
```

```
else:
```

```
    sell_shipping_cost = 0
```

```
# Display the invoice
```

```

    write.generate_and_display_sell_invoice(sell_customer_name, laptop_list,
sell_total_invoice_amount, sell_phone_no, sell_shipping_cost)

```

```

    # Generate and write the invoice to a text file

```

```

    write.write_sell_invoice(sell_customer_name, laptop_list,
sell_total_invoice_amount, sell_phone_no, sell_shipping_cost)

```

```

def get_distributor_laptop_order_details():

```

```

    """

```

```

    This function customer takes the input from the distributor
    to fill the details and generate order invoice

```

```

    """

```

```

    print("")

```

```

    print("=====
=====")

```

```

    print("    TO GENERATE ORDER INVOICE PLEASE ENTER THE FOLLOWING
DETAILS")

```

```

    print("=====
=====")

```

```

    print("")

```

```

    distributor_name = input_validation("    ENTER DISTRIBUTOR NAME: ",str)

```

```

    distributor_phone_no = input_validation("    ENTER PHONE NUMBER: ",int)

```

```

    # A list is created to store order details.

```

```

    laptop_list = []

```

```

    while True:

```

```
distributor_laptop_brand_name = input_validation("  ENTER LAPTOP BRAND:
",str)

distributor_laptop_model_name = input_validation("  ENTER LAPTOP MODEL:
",str)

distributor_laptop_quantity = input_validation("  ENTER LAPTOP QUANTITY:
",int)

distributor_laptop_price = input_validation("  ENTER LAPTOP PRICE: $",int)


# Calculate gross amount of laptop added to the list

distributor_laptop_gross_amount = distributor_laptop_quantity *
distributor_laptop_price


# 13% VAT amount is calculated from the gross amount
distributor_VAT_amount = 0.13 * distributor_laptop_gross_amount


# Calculate total amount

distributor_total_amount = distributor_laptop_gross_amount +
distributor_VAT_amount


# Add laptop details to the list created.
laptop_list.append([distributor_laptop_brand_name,
                    distributor_laptop_model_name,
                    distributor_laptop_quantity,
                    distributor_laptop_price,
                    distributor_laptop_gross_amount,
                    distributor_VAT_amount,
                    distributor_total_amount])

"""
```

If the user want to order more loop the prompt

otherwise display the invoice.

"""

```
distributor_order_more_laptop = input_validation(" DO YOU WANT TO ORDER
MORE LAPTOPS? (YES/NO): ",bool)
```

```
if distributor_order_more_laptop:
```

```
    continue
```

```
else:
```

```
    break
```

```
# Calculate total amount in invoice
```

```
distributor_total_invoice_amount = sum(laptop[6] for laptop in laptop_list)
```

```
# Calculate total gross amount in invoice
```

```
distributor_laptop_gross_amount = sum(laptop[4] for laptop in laptop_list)
```

```
return distributor_name, laptop_list, distributor_laptop_gross_amount,
distributor_total_invoice_amount, distributor_phone_no
```

```
if __name__ == "__main__":
```

```
    get_order_calculation()
```

"""

Generation of purchase invoice by taking all the required details from the distributor and generate the invoice by doing required calculations

"""

```
def get_customer_laptop_sell_details():
```

```

"""

This function takes the input from the customer
to fill the details and generate sell invoice

"""

print("")

print("=====
=====")

print("    TO GENERATE ORDER INVOICE PLEASE ENTER THE FOLLOWING
DETAILS")

print("=====
=====")

print("")

sell_customer_name = input_validation("    ENTER CUSTOMER NAME: ", str)
sell_phone_no = input_validation("    ENTER PHONE NUMBER: ", int)

# Store the product sold to the customer.
laptop_list = []

while True:
    sell_laptop_brand_name = input_validation("    ENTER LAPTOP BRAND: ", str)
    sell_laptop_model_name = input_validation("    ENTER LAPTOP MODEL: ", str)
    sell_laptop_quantity = input_validation("    ENTER LAPTOP QUANTITY: ", int)
    sell_laptop_price = input_validation("    ENTER LAPTOP PRICE: $", int) # Use
float to handle decimal prices

    sell_gross_amount = sell_laptop_quantity * sell_laptop_price
    laptop_list.append([sell_laptop_brand_name,

```

```
        sell_laptop_model_name,
        sell_laptop_quantity,
        sell_laptop_price,
        sell_gross_amount])

    sell_more_laptop = input_validation(" DO YOU WANT TO PURCHASE MORE
LAPTOPS? (YES/NO): ", bool)

    """

    If the user wants to order more, loop the prompt
    otherwise display the invoice.

    """

    if sell_more_laptop:
        continue
    else:
        break

# Calculate the total invoice amount
sell_total_invoice_amount = sum(product[4] for product in laptop_list)

# Assume the initial shipping cost is 0
sell_shipping_cost = 0

    return sell_customer_name, laptop_list, sell_total_invoice_amount, sell_phone_no,
    sell_shipping_cost

# Call the function to calculate the total amount and generate the invoice
if __name__ == "__main__":
    get_sell_calculation()
```

6.0.3 Appendix : Code for read.py module

```
# Read the text file laptop info.txt
open_textfile_path = "laptop info.txt"

with open(open_textfile_path,'r') as laptop_details_file:
    laptop_list = [line.strip().split(",") for line in laptop_details_file]

def display_laptop_list_table():
    """
    Function display_laptop_list_table()
    is declared to display the products in
    tabular format
    """
    header_laptop_details_table =
["S.N", "MODEL", "BRAND", "PRICE", "QUANTITY", "PROCESSOR", "GRAPHICS"]
    """
    Display the header of the table that includes
    S.N, MODEL, BRAND, PRICE, QUANTITY, PROCESSOR, GRAPHICS
    details of a laptop in organized order.
    """
    print(".....")
    print("")

    # Display the header of the table.
    print("+ " + "-" * 10 + "+" + "-" * 24 + "+" + "-" * 16 + "+" + "-" * 12 + "+" + "-" * 12 + "+" +
    "-" * 20 + "+" + "-" * 14 + "+")

    print("|{: ^10}|{: ^24}|{: ^16}|{: ^12}|{: ^12}|{: ^20}|{: ^14}|".format(*header_laptop_details_table))
```

```
print("+ " + "-" * 10 + "+" + "-" * 24 + "+" + "-" * 16 + "+" + "-" * 12 + "+" + "-" * 12 + "+" +
      "-" * 20 + "+" + "-" * 14 + "+")
```

Display each row of the table in a formatted order with the list of laptops and other details.

```
for laptops_table in laptop_list:
```

```
    print("{:^10}{:^24}{:^16}{:^12}{:^12}{:^20}{:^14}".format(*laptops_table))
```

Display the footer of the table.

```
print("+ " + "-" * 10 + "+" + "-" * 24 + "+" + "-" * 16 + "+" + "-" * 12 + "+" + "-" * 12 + "+" +
      "-" * 20 + "+" + "-" * 14 + "+")
```

```
print("")
```

```
print(".....")
```

```
def laptop_quantity_update():
```

```
    """
```

Function laptop_quantity_update()

is declared to check and update the quantity

of the laptops when the laptop is ordered

or sold by the store to its customer.

```
    """
```

while True:

```
    print(".....")
```

```
    input_index_of_laptop = input(" ENTER THE INDEX OF LAPTOP: ")
```

```
    print(".....")
```

```
    if not input_index_of_laptop.strip():
```

```
        """
```

Check if the index of the laptop entered by the user is empty or not.

If the field is empty, it displays an error message.

It loop back to the prompt and ask index of laptop.

```

"""

print(".....")
print("")
print("    ERROR !!!")
print("    EMPTY INDEX")
print("    PLEASE ENTER VALID LAPTOP INDEX.")
print("")
print(".....")
continue

```

if not input_index_of_laptop.isdigit():

```

"""

Check if the index of the laptop entered by the user is a string or not.
If the field has a string value, it displays an error message.
It loop back to the prompt and ask index of laptop.

"""

print(".....")
print("")
print("    ERROR !!!")
print("    INVALID INDEX")
print("    PLEASE ENTER A VALID NUMERIC LAPTOP INDEX.")
print("")
print(".....")
continue

```

index_of_laptop = int(input_index_of_laptop)

```

if index_of_laptop < 1 or index_of_laptop > len(laptop_list):
    """
    If the index of the laptop is less than 1 or greater than
    the number of laptops displayed on the screen, it validates and displays
    an error message as per the input from the user and loop back to the prompt.
    """

    print(".....")
    print("")
    print("    ERROR !!!")
    print("    INVALID LAPTOP INDEX.")
    print("")
    print(".....")
    continue

break

while True:
    print(".....")
    quantity_laptop = input("    ENTER QUANTITY:")
    print(".....")

    if not quantity_laptop.strip():
        """
        Check if the index of the laptop entered by the user is empty or not.
        If the field is empty, it displays an error message.
        It loop back to the prompt and ask index of laptop.
        """

        print(".....")

```

```

    print("")
    print("    ERROR !!!")
    print("    EMPTY INPUT.")
    print("    PLEASE ENTER VALID QUANTITY.")
    print("")
    print(".....")
    continue

if not quantity_laptop.isdigit():
    """
    Check if the quantity of the laptop entered by the user is a string or not.
    If the field has a string value, it displays an error message.
    It loop back to the prompt and ask quantity of laptop.
    """
    print(".....")
    print("")
    print("    ERROR !!!")
    print("    EMPTY INPUT.")
    print("    PLEASE ENTER VALID NUMERIC QUANTITY.")
    print("")
    print(".....")
    continue

reduce_quantity = int(quantity_laptop)

if reduce_quantity < 0 or reduce_quantity == 0:
    """

```

*If the quantity is less than 0 then display
out of stock message and loop back to prompt again.*

"""

print(".....")

print("")

print(" SORRY !!!")

print(" THE LAPTOP YOU ARE LOOKING FOR IS OUT OF STOCK...")

print("")

print(".....")

continue

break

Update the quantity in laptop.txt text file.

*laptop_list[index_of_laptop - 1][4] = str(int(laptop_list[index_of_laptop - 1][4]) -
reduce_quantity)*

Writing and updating quantity in laptop.txt file

with open(open_textfile_path, 'w') as laptop_details_file:

for information_laptop in laptop_list:

laptop_details_file.write(",".join(information_laptop) + "\n")

print(".....")

print("")

print(" LAPTOP QUNATITY UPDATED IN THE TEXT FILE.")

print("")

print(".....")

Total amount calculation and display

```

user_laptop_selected = laptop_list[index_of_laptop - 1]
user_laptop_price = float(user_laptop_selected[3].replace("$", ""))
user_total_amount = reduce_quantity * user_laptop_price

print(".....")
print("    TOTAL AMOUNT : $" + str(user_total_amount))
print(".....")

while True:
    print(".....")
    user_want_more_laptop = input("  DO YOU WANT TO BUY MORE LAPTOPS?
(YES/NO): ")
    print(".....")

    if not user_want_more_laptop.strip():
        """
        Check if the input is empty or not
        If it is empty display error message
        and loop back to prompt.
        """

        print(".....")
        print("")
        print("    ERROR !!!")
        print("    EMPTY INPUT.")
        print("    PLEASE ENTER 'YES' OR 'NO' ")
        print("")
        print(".....")
        continue

```

```

if user_want_more_laptop.isdigit():
    """
    Check if the input is numeric value or not
    If it is numeric value display error message
    and loop back to prompt.
    """

    print(".....")
    print("")
    print("    ERROR !!!")
    print("    INVALID INPUT.")
    print("    PLEASE ENTER 'YES' OR 'NO' ")
    print("")
    print(".....")
    continue

user_choice = user_want_more_laptop.lower()
"""
Ask customer if they want to buy more laptop.
If yes loop the prompt
If no display thank you message and other details.
"""

if user_choice == "yes":
    laptop_quantity_update()

elif user_choice == "no":
    print(".....")
    print("")
    print("    THANK YOU FOR SHOPPING WITH US !")

```

```

    print("          VISIT AGAIN !!!")
    print("")
    print(".....")

```

```

else:

```

```

    print(".....")
    print("")
    print("    ERROR !!!")
    print("    INVALID INPUT")
    print("    PLEASE ENTER 'YES' OR 'NO' ")
    print("")
    print(".....")
    continue
break

```

```

if __name__ == "__main__":
    laptop_quantity_update()

```

6.0.4 Appendix : Code for write.py module

```
import datetime

"""
    Date and time for invoice
    which is displayed at the time
    while ordering or selling laptop

    Displayed in dd/mm/yy and hh:mm:ss
    format.
"""

now = datetime.datetime.now()
string_day = str(now.day)
string_month = str(now.month)
string_year = str(now.year)
string_hour = str(now.hour)
string_minute = str(now.minute)
string_second = str(now.second)

string_date_time = string_day + "/" + string_month + "/" + string_year + " " + string_hour
+ ":" + string_minute + ":" + string_second

# Function to display order invoice
def generate_and_display_order_invoice(distributor_name, laptop_list,
distributor_laptop_gross_amount, distributor_total_invoice_amount,
distributor_phone_no):

print("=====
=====")
```



```

    print("")
    print("                KARKI ELECTRONICS AND LAPTOP SUPPLIERS")
    print("                KAPAN, KATHMANDU | CONTACT: 01-4270603")
    print("")

    print("=====
=====")

    print("                ORDER INVOICE")

    print("=====
=====")

    print("Name of customer: " + str(distributor_name) + "\n")
    print("Phone no: " + str(distributor_phone_no) + "\n")
    print("Date: " + str(string_date_time) + "\n")

    print("=====
=====")

    if laptop_list:
        print("-----
-----")
        print("{: <15}{: <15}{: <15}{: <15}{: <15}".format("LAPTOP BRAND", "LAPTOP
MODEL", "QUANTITY", "UNIT PRICE", "AMOUNT"))
        print("-----
-----")

        for laptop in laptop_list:
            print("{: <15}{: <15}{: <15}{: <15}{: <15}".format(laptop[0], laptop[1], laptop[2], "$" +
str(laptop[3]), "$" + str(laptop[4])))

        print("-----
---")

```

```

print("{:<30}Gross Amount: ${}\n".format("", distributor_laptop_gross_amount ))
print("{:<30}13% VAT: ${}\n".format("", sum(product[5] for product in laptop_list)))
print("{:<30}Total Amount: ${}\n".format("", distributor_total_invoice_amount))

print("=====
=====")

print("")
print("                THANK YOU FOR SHOPPING WITH US !!!")
print("                VISIT AGAIN !!!")
print("")

print("=====
=====")

# Function to write order invoice to .txt file
def write_order_invoice(distributor_name, laptop_list, distributor_laptop_gross_amount,
distributor_total_invoice_amount, distributor_phone_no):
    filename = f"{distributor_name}_order_invoice.txt"
    with open(filename, "w") as file:

file.write("=====
=====\\n")

    file.write("\\n")
    file.write("                KARKI ELECTRONICS AND LAPTOP SUPPLIERS
\\n")
    file.write("                KAPAN, KATHMANDU | CONTACT: 01-4270603
\\n")
    file.write("\\n")

file.write("=====
=====\\n")

    file.write("                ORDER INVOICE\\n")

```

```

file.write("=====
=====\\n")

file.write("Name of customer: " + str(distributor_name) + "\\n")
file.write("Phone no: " + str(distributor_phone_no) + "\\n")
file.write("Date: " + str(string_date_time) + "\\n")

file.write("=====
=====\\n")

if laptop_list:
    file.write("-----
-----\\n")

    file.write("{:<15}{:<15}{:<15}{:<15}{:<15}\\n".format("LAPTOP BRAND", "LAPTOP
MODEL", "QUANTITY", "UNIT PRICE", "AMOUNT"))

    file.write("-----
-----\\n")

    for laptop in laptop_list:
        file.write("{:<15}{:<15}{:<15}{:<15}{:<15}\\n".format(laptop[0], laptop[1],
laptop[2], "$" + str(laptop[3]), "$" + str(laptop[4])))

    file.write("-----
-----\\n")

    file.write("{:<30}      Gross Amount:    ${}\\n".format("",
distributor_laptop_gross_amount))

    file.write("{:<30}      13% VAT:        ${}\\n".format("", sum(product[5] for product
in laptop_list)))

    file.write("{:<30}      Total Amount:    ${}\\n".format("",
distributor_total_invoice_amount))

file.write("=====
=====\\n")

file.write("\\n")

```

```

file.write("                THANK YOU FOR SHOPPING WITH US !!!\n")
file.write("                VISIT AGAIN !!!\n")
file.write("\n")

```

```

file.write("=====
=====\\n")

```

```

# Function to display sell invoice

```

```

def generate_and_display_sell_invoice(sell_customer_name, laptop_list,
sell_total_invoice_amount, sell_phone_no, sell_shipping_cost):

```

```

    """

```

```

    Function to generate and display the sell invoice

```

```

    """

```

```

print("=====
=====")

```

```

    print("")

```

```

    print("                KARKI ELECTRONICS AND LAPTOP SUPPLIERS
")

```

```

    print("                KAPAN, KATHMANDU | CONTACT: 01-4270603
")

```

```

    print("")

```

```

print("=====
=====")

```

```

    print("                SELL INVOICE")

```

```

print("=====
=====")

```

```

    print("CUSTOMER NAME:", sell_customer_name)

```

```

print("PHONE NUMBER:", sell_phone_no)
print("Date: " + str(string_date_time) + "\n")
print("")

print("=====
=====")

if laptop_list:
    print("-----")
    print("{:<15}{:<15}{:<15}{:<15}{:<15}".format("LAPTOP BRAND", "LAPTOP
MODEL", "QUANTITY", "UNIT PRICE", "AMOUNT"))
    print("-----")
    for laptop in laptop_list:
        print("{:<15}{:<15}{:<15}{:<15}{:<15}".format(laptop[0], laptop[1], laptop[2], "$" +
str(laptop[3]), "$" + str(laptop[4])))

    print("-----")
    print("GROSS AMOUNT:", "$" +
str(sell_total_invoice_amount))
    print("SHIPPING COST:", "$" + str(sell_shipping_cost))
    print("-----")
    print("TOTAL:", "$" + str(sell_total_invoice_amount +
sell_shipping_cost))

print("=====
=====")

print("")
print("THANK YOU FOR SHOPPING WITH US !!!")

```

```

print("                                VISIT AGAIN !!!")
print("")

print("=====
=====")

# Function to write sell invoice to .txt file

def write_sell_invoice(sell_customer_name, laptop_list, sell_total_invoice_amount,
sell_phone_no, sell_shipping_cost):

    filename = f"{sell_customer_name}_sell_invoice.txt"

    with open(filename, "w") as file:

file.write("=====
=====\\n")

        file.write("\\n")

        file.write("                                KARKI ELECTRONICS AND LAPTOP SUPPLIERS
\\n")

        file.write("                                KAPAN, KATHMANDU | CONTACT: 01-4270603
\\n")

        file.write("\\n")

file.write("=====
=====\\n")

        file.write("                                SELL INVOICE\\n")

file.write("=====
=====\\n")

        file.write("CUSTOMER NAME: " + sell_customer_name + "\\n")
        file.write("PHONE NUMBER: " + sell_phone_no + "\\n")
        file.write("Date: " + str(string_date_time) + "\\n")
        file.write("\\n")

```

```

file.write("=====
=====\\n")

    if laptop_list:

        file.write("-----
-----\\n")

        file.write("{:<15}{:<15}{:<15}{:<15}{:<15}\\n".format("LAPTOP BRAND", "LAPTOP
MODEL", "QUANTITY", "UNIT PRICE", "AMOUNT"))

        file.write("-----
-----\\n")

        for laptop in laptop_list:

            file.write("{:<15}{:<15}{:<15}{:<15}{:<15}\\n".format(laptop[0], laptop[1],
laptop[2], "$" + str(laptop[3]), "$" + str(laptop[4])))

            file.write("-----
-----\\n")

            file.write("                                GROSS AMOUNT: $" +
str(sell_total_invoice_amount) + "\\n")

            file.write("                                SHIPPING COST: $" +
str(sell_shipping_cost) + "\\n")

            file.write("-----
-----\\n")

            file.write("                                TOTAL:    $" +
str(sell_total_invoice_amount + sell_shipping_cost) + "\\n")

file.write("=====
=====\\n")

file.write("\\n")

file.write("                                THANK YOU FOR SHOPPING WITH US !!!\\n")

```

```
file.write("VISIT AGAIN !!!\n")
```

```
file.write("\n")
```

```
file.write("=====  
=====\\n")
```