



FUNDAMENTALS OF COMPUTING

60% Individual Coursework

2022-23 Autumn

Student Name: Krish Bhattarai

London Met ID: 22067570

College ID: NP01CP4A220071

Assignment Due Date: Friday, May 12, 2023

Assignment Submission Date: Wednesday, May 10, 2023

Word Count: 3740

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.

Contents

1. Introduction	1
Introduction to the Project.....	1
2. Algorithm.....	2
Algorithm of this program.....	2
Flowchart	4
3. Pseudocode	6
Pseudocode of main.py	6
Pseudocode for read.py.....	10
Pseudocode for operation.py.....	11
Pseudocode for write.py	20
4. Data Structure	29
1. Primitive Data Structure	29
Integers	29
String.....	30
Boolean.....	30
2. Non- Primitive Data Structure.....	31
Lists.....	31
Dictionary	31
3. Program	32
a. Main.py	32
b. Operations.py.....	32
c. Read.py	32

d. Write.py	32
e. Items.py	33
SELLING Laptops in the program.....	33
PURCHASING laptops in the program	36
Termination.....	39
5. Testing	40
Test 1 - Implementation of Try, except.	40
Test 2 – Select purchase and provide negative value as input.	41
Test 3-Select the non-existent value as input.	42
Test 4 – Test the complete purchase process and show the purchased laptops in a text file.	44
Test 5 – Test the complete sell process and show the sold laptops in a text file.	47
Test 6 – Test the quantity being added while purchasing the laptop.	51
Test 7 – Test the quantity being deducted while selling the laptop.	52
6. Conclusion	53
7. Appendix.....	55
Code for Main.py	55
Code for Operations.py.....	61
Code for read.py	68
Code for Write.py.....	69

Table of Figures

Figure 1: FLOWCHART	4
Figure 2: FLOWCHART	5
Figure 3: Implementation of Integer	29
Figure 4: Implementation of String	30
Figure 5: Implementation of Boolean.....	30
Figure 6: Implementation of Lists	31
Figure 7: Implementation of Dictionary.....	31
Figure 8: Selling of Laptop	33
Figure 9: Selling of Laptop	33
Figure 10: Bill generation of Selling of Laptop.....	34
Figure 11: File Generation.....	34
Figure 12: Writing of bill in text file.....	35
Figure 13: Buying of Laptops	36
Figure 14: Buying of Laptops	36
Figure 15: Bill Generation of Purchasing Laptops.	37
Figure 16: Creation of Text file	37
Figure 17: Writing of bill in text file.....	38
Figure 18: Terminating the program.	39
Figure 19: Testing the Implementation of Try, Except.....	40
Figure 20: Testing the Implementation of Try, Except.....	41
Figure 21: Providing negative value as input.....	42
Figure 22: Providing non- existent value as input.....	43
Figure 23: Testing the complete purchase process.....	44

Figure 24: Testing the complete purchase process.....	45
Figure 25: Testing the complete purchase process.....	45
Figure 26: Testing the complete purchase process.....	46
Figure 27: Testing the complete purchase process.....	46
Figure 28: Test the selling process.....	48
Figure 29: Testing the selling process.....	48
Figure 30: Test the selling process.....	49
Figure 31: Testing the selling process.....	50
Figure 32: Testing the selling process.....	50
Figure 33: Testing the quantity being added while purchasing the laptop.	51
Figure 34: Testing the quantity being added while purchasing the laptop.	51
Figure 35: Testing the quantity being deducted while selling the laptop.....	52
Figure 36: Testing the quantity being deducted while selling the laptop.....	52

Table of Tables

Table 1: Testing the implementation of Try, Except.	40
Table 2: Providing negative value as input.	41
Table 3: Provide non- existent value as input.	42
Table 4: Testing the complete purchase process.	44
Table 5: Testing the selling process.	47
Table 6: Testing the quantity being added while purchasing the laptop.	51
Table 7: Testing the quantity being deducted while selling the laptop.	52

1. Introduction

Python is a high-level and object-oriented program with built in data structure namely lists, dictionaries, sets, etc. which makes writing complex programs quick and easy. Syntax in python is easy to learn making the code readable and understandable by others. Codes in python can be reused thanks to 'module and packages' features promoting programmers to reuse the code making the development process easy. Python can be used to analyse data, develop websites, develop website, etc.

Python provides various operators like arithmetic operators, logical operators, comparison operators, et. The arithmetic operators are used to perform mathematical calculations such as additions, subtractions, multiplication, division, etc. The logical operator evaluates a single expression by combining multiple conditions. Comparison operator returns Boolean value (True or False) by comparing values and variables.

Python is frequently used to build the back end of websites and applications, which consists of parts that are not visible to the users. The tasks carried out can be, communicating with databases, transferring, and processing data to and from the servers, and ensuring security (Rossum, 2023).

Introduction to the Project

The development of a program that can manage information about available laptops The project requires students to develop a program that manages the purchase and sale of laptops. The program allows customers to purchase laptops from the laptop store. When a laptop is sold, the program records the transaction as the username, phone number, address laptop name, company, name, quantity, and price. Shipping cost is later added to the customers' purchase. An invoice is then generated to keep track of the order. The sold laptop is then updated in the text file named 'Items.txt'. The invoice created is named as the username and the time of purchase.

The program also has the ability to order laptops from the distributor. When a laptop is ordered, the program records the transaction as the distributor name, laptop name,

company, name, quantity, and price. VAT is later added to the purchase. An invoice is generated making it easy to track the orders. The purchased quantity is then updated in the text file named 'Items.txt'. The invoice created is named as the distributor and the time of purchase.

2. Algorithm

Algorithm is a logical representation of a program which is created before writing the actual program. Algorithms are developed independently meaning, can be applied in various programming languages such as python, Java, C++, etc. It is a step-by-step process or a set of instructions that is used for solving a problem or completing a task (Simplilearn, 2023).

Algorithm of this program

- Step 1: START
- Step 2: Display welcome message and shop information.
- Step 3: Obtain desired input from the user.
- Step 4: If user_input is 1, display the laptop details for Sell.
- Step 5: If user_input is 2, display the laptop details for Buy.
- Step 6: If user_input is 3, Exit the program.
- Step 7: Validate the given integer.
- Step 8: Validate the name, phone number and the address provided.
- Step 9: Read Items.txt file.
- Step 10: Validate the provided serial number.
- Step 11: Validate the provided number of laptops to purchase.

- Step 12: Validate if the user wants to purchase more.
- Step 13: If user_request = Y, Step 9.
- Step 14: If user_request = N, Print Bill.
- Step 15: Write Bill.
- Step 16: If user_input = 2, display the laptop details to Buy.
- Step 17: Validate the Distributor's Name.
- Step 18: Read Items.txt file.
- Step 19: Validate the Provided serial Number.
- Step 20: Provide the number of laptops to purchase.
- Step 21: If user_request = Y, Step 18.
- Step 22: If user_request = N, Print Bill.
- Step 23: Write Bill.
- Step 24: If user_input = 3, Exit.
- Step 25: END

Flowchart

Flowchart provides a visual representation of data that shows the flow of a program. It helps coders understand the development process visually. Flowchart optimizes the operation, making the development process simple. It doesn't contain the most detail but is an effective way to convey the logic of the program (GeeksforGeeks, 2023).

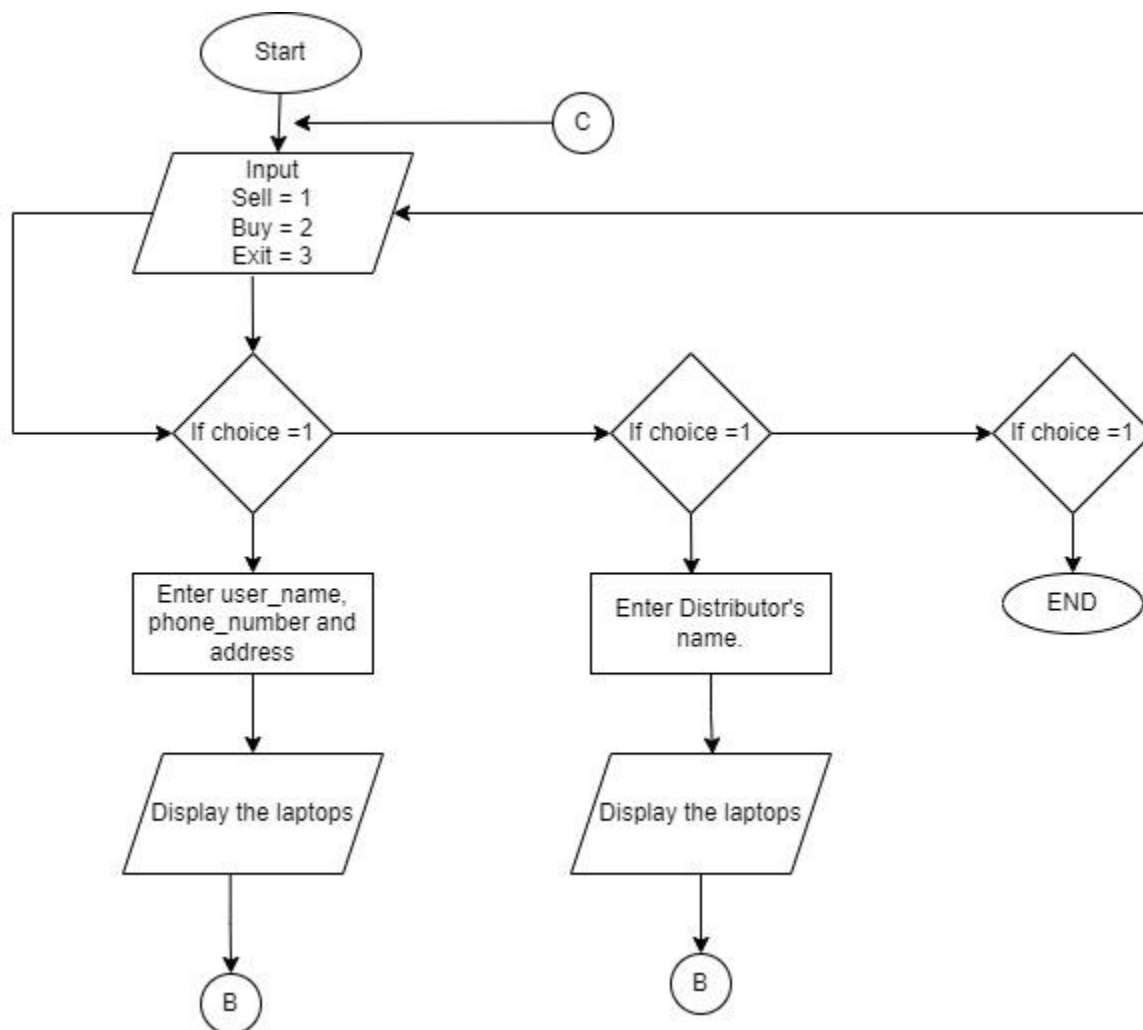


Figure 1: FLOWCHART

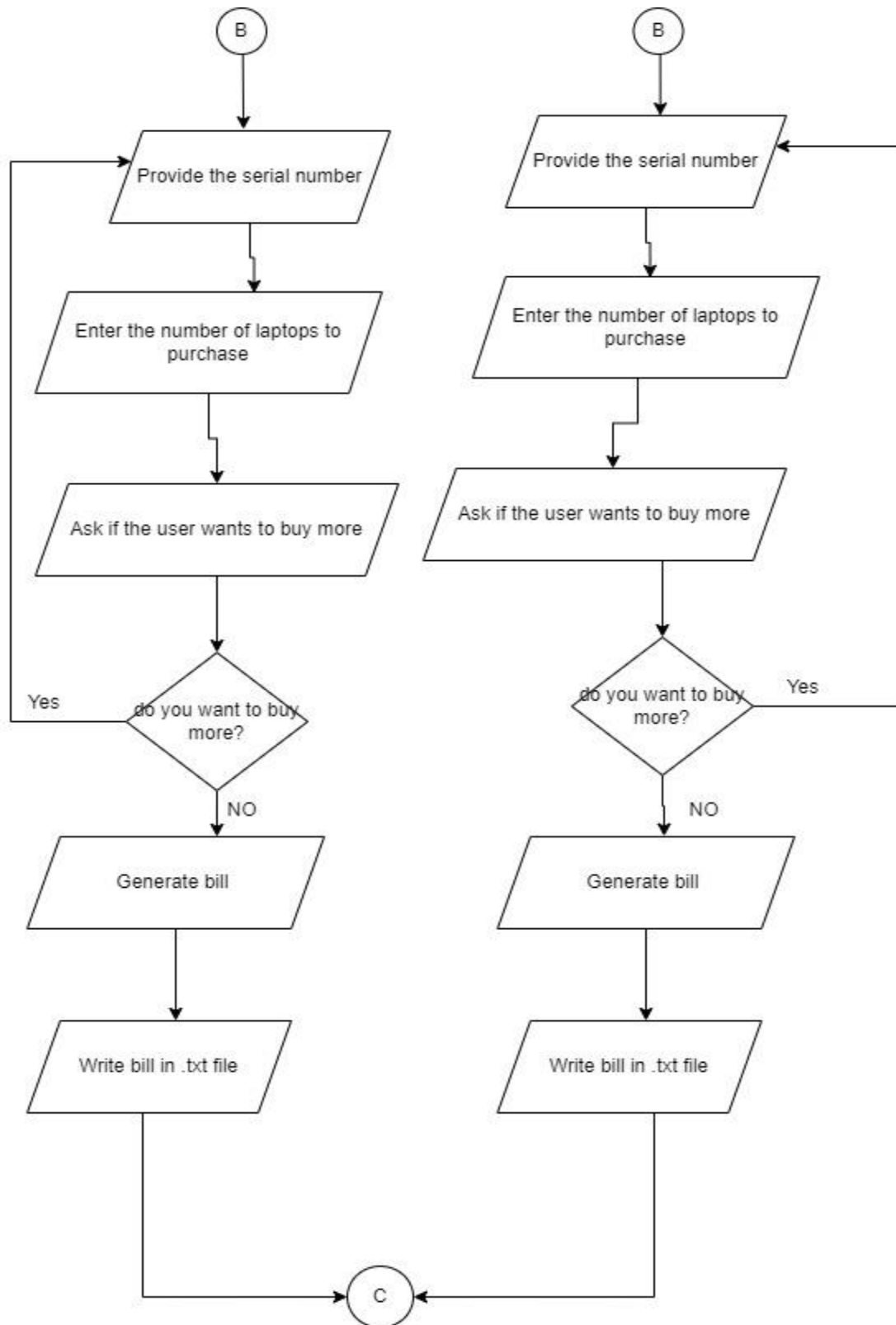


Figure 2: FLOWCHART

3. Pseudocode

Pseudocode also known as 'false code' or 'fake code' is an informal way of writing any program that humans can easily understand. It includes few details about the program but helps summarize it. Pseudocode is written in such a way that the program can be understood by virtually anyone. This helps clarify how each line of the program works, which makes the code construction phase easier for the programmer (GeeksforGeeks, 2023).

Pseudocode of main.py

IMPORT read, write, operation, datetime

PRINT welcome message

OPEN 'Items.txt' in read mode

CREATE an empty dictionary laptop_dictionary

INITIALIZE laptop_id to 1

FOR each line in file:

REMOVE "\n" from the line

STORE result as a list of laptop attributes by splitting line by comma

 ..

 ..

CALL laptop_dictionary = laptops()

SET loop = True

WHILE loop is True

PRINT "1 to SELL"

PRINT "2 to BUY"

PRINT "3 to EXIT"

WHILE True

TRY to

ENTER an integer and assign it to the variable user_input

IF user_input <= 0 or > 3

RAISE value error with the message "Please provide a valid number"

BREAK out of the loop if the user_input is valid

IF ValueError is encountered

PRINT "Please provide a valid number"

IF user_input is 1

CALL sell() function and assign the returned values to the variables user_name, phone_number and user_address

CALL plaptop() function and assign the returned values to the variable purchased_laptops

SET buy_more to True

WHILE buy_more is True

CALL table1()

CALL items()

CALL serial_no = serial()

WHILE serial_no <=0 or serial_no is> the length of laptop_dictionary

WHILE True

IF serial_no.isalpha()

PRINT("Please enter an available serial number.")

ELSE

BREAK out of the loop if the serial number is valid

CALL serial_no = checkserial(laptop_dictionary)

CALL selected_laptop_quantity = valid(laptop, dictionary, serial_no)

CALL update(laptop_dictionary, serial_no, user_quantity)

CALL product_name, unit_price, total_price, company_name =
get(laptop_dictionary, serial_no, user_quantity)

APPEND purchased_laptops.append([product_name, company_name,
selected_laptop_quantity, unit_price, total_price])

CALL shipping_cost, user_request = more()

IF user_request is 'Y'

SET buy_more to True

ELSE

SET total = 0

FOR each purchased laptops in purchased_laptops

ADD total+=int(i[4])

SET sum_total to total + shipping_cost

CALL printsell(now, user_name, phone_number, user_address,
shipping_cost, sum_total, purchased_laptops)

CALL sellbill(user_name, date_time, now, phone_number,
user_address, purchased_laptops, shipping_cost, sum_total)

SET buy_more to False

ELIF user_input is 2

WHILE True

TRY to

ENTER distributor_name

IF distributor_name is empty

PRINT "Textfield is empty."

ELSE

BREAK out of the loop

CALL plaptop() function and assign the returned values to the variable
purchased_laptops

SET buy_more to True

WHILE buy_more is True

CALL table2()

CALL items()

CALL serial_no = serial2()

WHILE serial_no <=0 or serial_no is > the length of laptop_dictionary

```
    CALL user_quantity, serial_no = checkserial2()

    CALL selected_laptop_quantity = valid2(laptop, dictionary,
    serial_no)

    CALL update2 (laptop_dictionary, serial_no, user_quantity)

    CALL product_name, unit_price, total_price, item_price,
    company_name = get2(laptop_dictionary, serial_no, user_quantity,
    purchased_laptops)

    CALL user_request = more2()

    IF user_request is "Y"

        SET but_more to True

    ELSE

        SET buy_more to False

        CALL printbuy(now, distributor_name, purchased_laptops)

        CALL buybill(distributor_name, date_time, now,
        purchased_laptops)
```

Pseudocode for read.py

```
DEFINE laptops()

    OPEN Items.txt in read mode

    CREATE laptop_dictionary = {}

    INITIALIZE laptop_id = 1
```


FOR line in file:

REPLACE line = line.replace("\n", "")

SPLIT laptop_dictionary.update({laptop_id: line.split(",")})

INCREASE laptop_id = laptop_id + 1

CLOSE

Pseudocode for operation.py

IMPORT datetime

IMPORT all from write

DEFINE sell():

WHILE True:

INPUT user_name = input("Please enter your name: ")

IF user_name == "":

PRINT("Empty Textfield.")

ELSE:

BREAK

PRINT ("\n")

WHILE True:

INPUT phone_number = (input("Please enter your phone no: "))

IF phone_number.isdigit():

INPUT phone_number = int(phone_number)

BREAK

ELIF phone_number == "":

PRINT ("Please enter a valid phone number!!!")

ELSE:

PRINT ("Please enter a valid phone number!!!")

PRINT ("\n")

WHILE True:

INPUT user_address = input("Please enter your Address: ")

IF user_address == "":

PRINT ("Empty Textfield.")

ELSE:

BREAK

RETURN user_name, phone_number, user_address

DEFINE plaptop():

 purchased_laptops = []

RETURN purchased_laptops

DEFINE table1():

PRINT ("-----
-----")

PRINT ("S.N \t\t Laptop Name \t\t Company Name \t\t Price \t\t Quantity \t Processor
\t\t GPU")

PRINT ("-----
-----")

DEFINE serial():

INPUT serial_no = (input("Provide the serial number of the laptop you want to purchase:
"))

PRINT ("\n")

WHILE not serial_no.isdigit():

INPUT serial_no = input("Please enter a valid serial number: ")

serial_no = int(serial_no)

RETURN serial_no

#Valid ID

DEFINE checkserial():

while serial_no <= 0 or serial_no > len (laptop_dictionary):

PRINT ("Please select an available Serial N.O.")

PRINT ("\n")

INT serial_no = int(input("Provide the serial number of the laptop you want to purchase:"))

PRINT ("\n")

RETURN serial_no

#Valid Quantity

DEFINE valid(laptop_dictionary, serial_no):

selected_laptop_quantity = laptop_dictionary[serial_no][3]

RETURN selected_laptop_quantity

DEFINE change_quantity(selected_laptop_quantity):

WHILE True:

TRY:

```
INPUT user_quantity = input("Enter the number of laptops you want to purchase:")
```

```
IF not user_quantity:
```

```
    RAISE ValueError("Please provide a value")
```

```
user_quantity = int(user_quantity)
```

```
IF user_quantity <= 0 or user_quantity > int(selected_laptop_quantity):
```

```
    RAISE ValueError("The quantity you are looking for is unavailable to purchase.")
```

```
RETURN user_quantity
```

```
EXCEPT ValueError:
```

```
    PRINT ("Invalid input: ")
```

```
#Update the text file
```

```
DEFINE update(laptop_dictionary, serial_no, user_quantity):
```

```
    laptop_dictionary[serial_no][3] = int(laptop_dictionary[serial_no][3]) - int(user_quantity)
```

```
    dictionary(laptop_dictionary)
```

```
#getting user purchased items
```

```
DEFINE get(laptop_dictionary, serial_no, user_quantity, purchased_laptops):
```

```
    product_name = laptop_dictionary[serial_no][0]
```

```
selected_laptop_quantity = user_quantity

unit_price = laptop_dictionary[serial_no][2]

item_price = laptop_dictionary[serial_no][2].replace("$", "")

total_price = int(item_price)*int(selected_laptop_quantity)

company_name = laptop_dictionary[serial_no][1]
```

```
    APPEND    purchased_laptops.append([product_name,    company_name,
selected_laptop_quantity, unit_price, total_price])
```

```
    RETURN product_name, unit_price, total_price, company_name
```

```
DEFINE more():
```

```
    INPUT user_request = input("Would you like to purchase more? If yes press Y. If no
press any other button.").upper()
```

```
    PRINT ("\n")
```

```
    INPUT shipping_cost = 200
```

```
    RETURN shipping_cost, user_request
```

#BUY#BUY#BUY#BUY#BUY

#BUY#BUY#BUY#BUY#BUY

#BUY#BUY#BUY#BUY#BUY

DEFINE table2 ():

```
    PRINT ("-----  
-----")
```

```
    PRINT ("S.N \t\t Laptop Name \t\t Company Name \t\t Price \t\t Quantity \t Processor  
\t\t GPU")
```

```
    PRINT ("-----  
-----")
```

DEFINE serial2 ():

```
    INPUT serial_no = (input("Provide the serial number of the laptop you want to purchase:  
"))
```

```
    PRINT ("\n")
```

```
    WHILE not serial_no.isdigit():
```

```
        INPUT serial_no = input("Please enter a valid serial number: ")
```

```
    serial_no = int(serial_no)
```

```
    RETURN serial_no
```

#Valid ID

DEFINE checkserial2 ():

PRINT ("Please select an available Serial N.O.")

PRINT ("\n")

INPUT serial_no = int(input("Provide the serial number of the laptop you want to purchase: "))

PRINT ("\n")

#

RETURN serial_no

return user_quantity, serial_no

#Valid ID

#Valid Quantity

DEFINE valid2 (laptop_dictionary, serial_no):

WHILE True:

INPUT user_input = input("Enter the number of laptops you want to purchase: ")

IF user_input.strip() == "":

PRINT ("Empty input")

ELIF not user_input.isdigit():

PRINT ("Invalid input. Please enter a valid number.")

ELSE:

user_quantity = int(user_input)

BREAK

selected_laptop_quantity = laptop_dictionary[serial_no][3]

RETURN selected_laptop_quantity, user_quantity

DEFINE update2(laptop_dictionary, serial_no, user_quantity):

laptop_dictionary[serial_no][3] = int(laptop_dictionary[serial_no][3]) +
int(user_quantity)

dictionary (laptop_dictionary)

#getting user purchased items

DEFINE get2(laptop_dictionary, serial_no, user_quantity, purchased_laptops):

product_name = laptop_dictionary[serial_no][0]

selected_laptop_quantity = user_quantity

unit_price = laptop_dictionary[serial_no][2]

item_price = laptop_dictionary[serial_no][2].replace("\$", "")

```
total_price = int(item_price)*int(selected_laptop_quantity)
```

```
company_name = laptop_dictionary[serial_no][1]
```

```
                purchased_laptops.append([product_name,                company_name,  
selected_laptop_quantity, unit_price, total_price])
```

```
RETURN product_name, unit_price, total_price, item_price, company_name
```

```
#getting user purchased items
```

```
DEFINE more2 ():
```

```
INPUT user_request = input("Would you like to buy more? If YES press Y. If NO press  
any other button.").upper() #Converts value into Upper Case
```

```
PRINT ("\n")
```

```
RETURN user_request
```

Pseudocode for write.py

```
DEFINE dictionary(laptop_dictionary):
```

```
OPEN "Items.txt" in write mode
```

```
FOR values in laptop_dictionary.values():
```

```
WRITE(str(values[0])+", "+str(values[1])+", "+str(values[2])+", "+str(values[3]  
)+", "+str(values[4])+", "+str(values[5]))
```

```
WRITE "\n"
```

CLOSE

DEFINE(user_name, date_time, now, phone_number, user_address, purchased_laptops, shipping_cost, sum_total):

CREATE filename = f"{user_name} {date_time}.txt"

OPEN open(filename, "w")

WRITE "\t\t\t\t\tKrish's Laptop Shop"

WRITE "\n"

WRITE "\t\t\t\t\tSales invoice"

WRITE "\n"

WRITE "Date/Time"+str(now),

WRITE "\n"

WRITE "Customer Name: "+str(user_name)

WRITE "\n"

WRITE "Customer Phone No: "+str(phone_number)

WRITE file.write("\n")

WRITE "Customer address: "+str(user_address)

WRITE "\n"

WRITE "-----
---" "\n"

WRITE "Laptop Name| \t\t |Company Name \t\t |Quantity \t\t |Price \t\t\t " " |Total| "
"\n"

```

WRITE "-----
---" "\n"

```

```

FOR i in purchased_laptops:

```

```

    WRITE str(i[0])+"\t  " +str(i[1])+" \t\t  "+"  "+str(i[2])+" \t  "+str(i[3]) +
    "\t\t\t  " + "$"+ str(i[4]) + "\n"

```

```

WRITE "-----
---" "\n"

```

```

WRITE "\n"

```

```

WRITE "Shipping Cost: "+ str(shipping_cost)

```

```

WRITE "\n"

```

```

WRITE "Sum Total: $"+str(sum_total)

```

```

WRITE "\n"

```

```

DEFINE buybill(distributor_name, date_time, now, purchased_laptops)

```

```

    TOTAL = 0

```

```

    FOR i in purchased_laptops:

```

```

        CALCULATE total+=int(i[4])

```

```

    CALCULATE sum_total = total

```

```

    CALCULATE vat = sum_total * (13/100)

```

```

    CALCULATE vat_amt = sum_total + vat

```

```

    CREATE filename = f"{distributor_name} {date_time}.txt"

```

OPEN filename in write mode

WRITE "\n"

WRITE "\n"

WRITE "\n"

WRITE "\n"

```
WRITE "\t\t\t\t\t" " " "Date/Time"+str(now),
```

WRITE "\n"

WRITE "\t\t\t\t\t Sasto Laptops | Phone No: 999999999"

WRITE "\n"

WRITE"-----"

WRITE “\n”

WRITE “\n”

```
WRITE "Distributor Name: " + str(distributor_name)
```

WRITE “\n”

WRITE “\n”

WRITE "-----"

WRITE “\n”

WRITE "Laptop Name| \t\t |Company Name \t\t |Quantity \t\t |Price \t\t |Total|"

WRITE “\n”

```

WRITE "-----"
"\n"

FOR "-----" "\n"

    WRITE (str(i[0])+"\t    " " "+str(i[1])+ " \t\t    " "+"    "+str(i[2])+ "\t\t
    "+str(i[3]) + "\t    "+"$"+str(i[4]) + "\n")

WRITE "\n"

WRITE "-----"

WRITE "\n"

WRITE "\n"

WRITE "Sum Total: $" +str(sum_total)

WRITE "\n"

WRITE "VAT amount: " +str(vat)

WRITE "\n"

WRITE "Total Amount with VAT: " +str(vat_amt)


DEFINE printsell(now, user_name, phone_number, user_address, shipping_cost,
sum_total, purchased_laptops):

    PRINT("\n")

    PRINT ("\t\t\t\t\tKrish's Laptop Shop")

    PRINT ("\n")

    PRINT ("\t\t\t\tBafal, Kathmandu | Phone No: 000000000000")

```

```
    PRINT ("\n")

    PRINT ("\t\t\t\t\t Sales invoice")

    PRINT ("\n")

    PRINT ("Date|Time: "+str(now),)

    PRINT ("Customer Name: "+str(user_name))

    PRINT ("Customer Phone No: "+str(phone_number))

    PRINT ("Customer Address: "+str(user_address))

    PRINT ("\n")

    PRINT ("-----")
    -----")

    PRINT ("Laptop Name| \t\t\t |Company Name \t\t\t |Quantity \t\t |Price \t\t\t |Total|")
    ")

    PRINT ("-----")
    -----")

    PRINT ("\n")

    FOR i in purchased_laptops:

        PRINT (i[0], "\t\t\t", i[1], "\t\t\t", "    ", i[2], "\t\t", i[3], "\t\t\t", "$", i[4], "\n")#, i[4],
        "\t\t\t", i[5])

    PRINT ("-----")
    -----")

    PRINT("Shipping Cost: "+ str(shipping_cost))

    PRINT ("Sum Total: $" +str(sum_total))
```

```
PRINT ("\n")
```

```
PRINT ("\n")
```

```
PRINT ("\n")
```

```
#BUY BILL
```

```
DEFINE printbuy(now, distributor_name, purchased_laptops):
```

```
    TOTAL= 0
```

```
    FOR i in purchased_laptops:
```

```
        total+=int(i[4])
```

```
    CALCULATE sum_total = total
```

```
    CALCULATE vat = sum_total * (13/100)
```

```
    CALCULATE vat_amt = sum_total + vat
```

```
PRINT ("\n")
```

```
PRINT ("\t\t\t\t\tSasto Laptop Store")
```

```
PRINT ("\n")
```

```
PRINT ("\t\t\t\t\tHattiban, Lalitpur | Phone No: 9999999999")
```

```
PRINT ("\n")
```

```
PRINT ("\t\t\t\t\tSales Receipt")
```

```
PRINT ("\n")
```



```

    PRINT ("-----
-----")

    PRINT ("\n")

    PRINT ("Print Date|Time: "+str(now),)

    PRINT ("\n")

    PRINT ("Customer Name: "+str(distributor_name))

    PRINT ("\n")

    PRINT ("Purchased products: ")

    PRINT ("\n")

    PRINT ("-----
-----")

    PRINT ("Laptop Name| \t\t\t |Company Name \t\t\t " " |Quantity \t\t |Price \t\t\t
|Total| ")

    PRINT ("-----
-----")

    PRINT ("\n")

    FOR i in purchased_laptops:

        PRINT (i[0], "\t\t\t", i[1], "\t\t\t", " ", i[2], "\t\t\t", i[3], "\t\t\t", "$", i[4], "\n")#, i[4],
        "\t\t\t", i[5])

    PRINT ("-----
-----")

    PRINT ("\n")

```

```
PRINT ("Sum Total: $" +str(sum_total))
```

```
PRINT ("\n")
```

```
PRINT ("VAT amount: " +str(vat))
```

```
PRINT ("\n")
```

```
PRINT ("Total amount with VAT Amount: " +str(vat_amt))
```

```
PRINT ("\n")
```

4. Data Structure

Data Structures are used to store and organize data so that the data can be accessed and worked with more efficiently. There are two types of data structure in python, they are mutable and immutable data structure. The mutable data structures can be changed or modified after they have been created, while the immutable data structures cannot be changed or modified after they have been created (Dataquest, 2023).

1. Primitive Data Structure

Integers

Integer is a primitive data structure that can only take numeric values. Integer represents whole numbers (DataCamp, 2023).

```
try:
    user_input = int(input("What do you want to do?"))
    if user_input <=0 or user_input > 3:
        raise ValueError("Please provide a valid number")
    break
except ValueError:
    print("please provide a valid number")
```

Figure 3: Implementation of Integer

String

String is a primitive data structure that can hold any set of characters like alphabets, numbers, symbols, etc. Strings are considered to be immutable, meaning that it cannot be changed or modified after its creation. The '+' operator can be used to join strings together (DataCamp, 2023).

```
def table1():  
    print("-----")  
    print("S.N \t\t Laptop Name \t\t Company Name \t\t Price \t\t Quantity \t Processor \t\t GPU")  
    print("-----")
```

Figure 4: Implementation of String

Boolean

Boolean are conditional operators such as (equal to) ==, (less than or equal to) <=, (not equal to) !=. They are comparison expressions that takes the values: True and False (DataCamp, 2023).

```
while True:  
    phone_number = (input("Please enter your phone no: "))  
    if phone_number.isdigit():  
        phone_number = int(phone_number)  
        break  
    elif phone_number == "":  
        print("Please enter a valid phone number!!!")  
    else:  
        print("Please enter a valid phone number!!!")  
  
print("\n")
```

Figure 5: Implementation of Boolean

2. Non- Primitive Data Structure

Lists

Lists store heterogeneous items that can change their content without changing their identity. Lists store elements in containers '[]' and are separated by comma ','. Lists are mostly used when the data needs to be accessed frequently that can be done using indexing (PythonGeeks, 2023).

```
def plaptop():  
    purchased_laptops = []  
  
    return purchased_laptops
```

Figure 6: Implementation of Lists

Dictionary

Dictionaries are data structures that consists of a key-value pairs. Dictionaries are defined by {}, the key value pair are separated by ':' and each key value pair are separated by ','. The keys should be unique and cannot be changed (DataCamp, 2023).

```
def laptops():  
    file = open("Items.txt", "r")  
    laptop_dictionary = {}  
    laptop_id = 1  
    for line in file:  
        line = line.replace("\n", "")  
        laptop_dictionary.update({laptop_id: line})  
        laptop_id += 1
```

Figure 7: Implementation of Dictionary

3. Program

The program consists of 5 files named main.py, operations.py, read.py, write.py and a text file named Items.py.

a. Main.py

The main.py is the main logic for the program. The program is run from this file. Operations, Read and Write are imported in main.py. It displays the welcome message for the laptop shop.

b. Operations.py

The operations.py contains all the codes to carry out the operations in the main file. There are 17 functions defined in operations.py. It handles validations and the arithmetic calculations in the program.

c. Read.py

Read.py reads the data from the text file. It contains all the information about the available laptops. It reads all the information about the available laptops in the text file. There are two functions defined in this file, 'laptops' and 'items'. The 'laptops' function reads the available laptops from Items.txt. The 'items' function prints the laptops with the serial number.

d. Write.py

The write.py contains all the function to print and write the bill. The function to generate bills is defined in write.py and then called in main.py. It contains the code for printing the bill in terminal and generating the bill in a text file. The write.py is imported in main.py so that can be called from write.

e. Items.py

The Items.txt contains information about all the available laptops in the laptop shop. Read.py reads the contents from Items.py and displays it in the terminal.

SELLING Laptops in the program

```
=====
Welcome to Laptop Shop
=====
Bafal, Kathmandu
=====
1 to Sell
2 to Purchase
3 to Exit
What do you want to do?
Please enter your name: Krish

Please enter your phone no: 9800000000

Please enter your Address: Sitapaila

=====
S.N      Laptop Name      Company Name      Price      Quantity      Processor      GPU
=====
1         Razer Blade      Razer             $2000      178           17 7th Gen     GTX 3060
2         XPS              Dell              $1976      82            15 9th Gen     GTX 3070
3         Alienware       Alienware         $1978      30            15 9th Gen     GTX 3070
4         Swift 7          Acer              $900       48            15 9th Gen     GTX 3070
5         Macbook Pro 16   Apple             $3500      60            15 9th Gen     GTX 3070
=====

Provide the serial number of the laptop you want to purchase: 1

Enter the number of laptops you want to purchase: 6
Would you like to purchase more? If yes press Y. If no press any other button.y
```

Figure 8: Selling of Laptop

```
Would you like to purchase more? If yes press Y. If no press any other button.y

=====
S.N      Laptop Name      Company Name      Price      Quantity      Processor      GPU
=====
1         Razer Blade      Razer             $2000      172           17 7th Gen     GTX 3060
2         XPS              Dell              $1976      82            15 9th Gen     GTX 3070
3         Alienware       Alienware         $1978      30            15 9th Gen     GTX 3070
4         Swift 7          Acer              $900       48            15 9th Gen     GTX 3070
5         Macbook Pro 16   Apple             $3500      60            15 9th Gen     GTX 3070
=====

Provide the serial number of the laptop you want to purchase: 3

Enter the number of laptops you want to purchase: 4
Would you like to purchase more? If yes press Y. If no press any other button.n
```

Figure 9: Selling of Laptop

```

Krish's Laptop Shop

Bafal, Kathmandu | Phone No: 000000000000

Sales invoice

Date|Time: 2023-05-09 13:11:23.142404
Customer Name: Krish
Customer Phone No: 9800000000
Customer Address: Sitapaila

-----
|Laptop Name|          |Company Name          |Quantity|Price  |Total|
|-----|
Razer Blade          Razer                6       $2000  $ 12000
Alienware             Alienware            4       $1978  $ 7912
-----

Shipping Cost: 200
Sum Total: $20112

1 to Sell
2 to Purchase
3 to Exit
What do you want to do|

```

Figure 10: Bill generation of Selling of Laptop

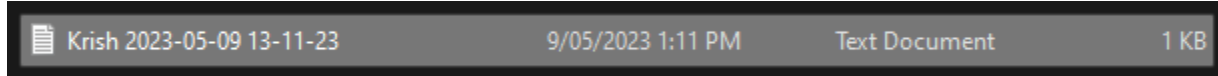
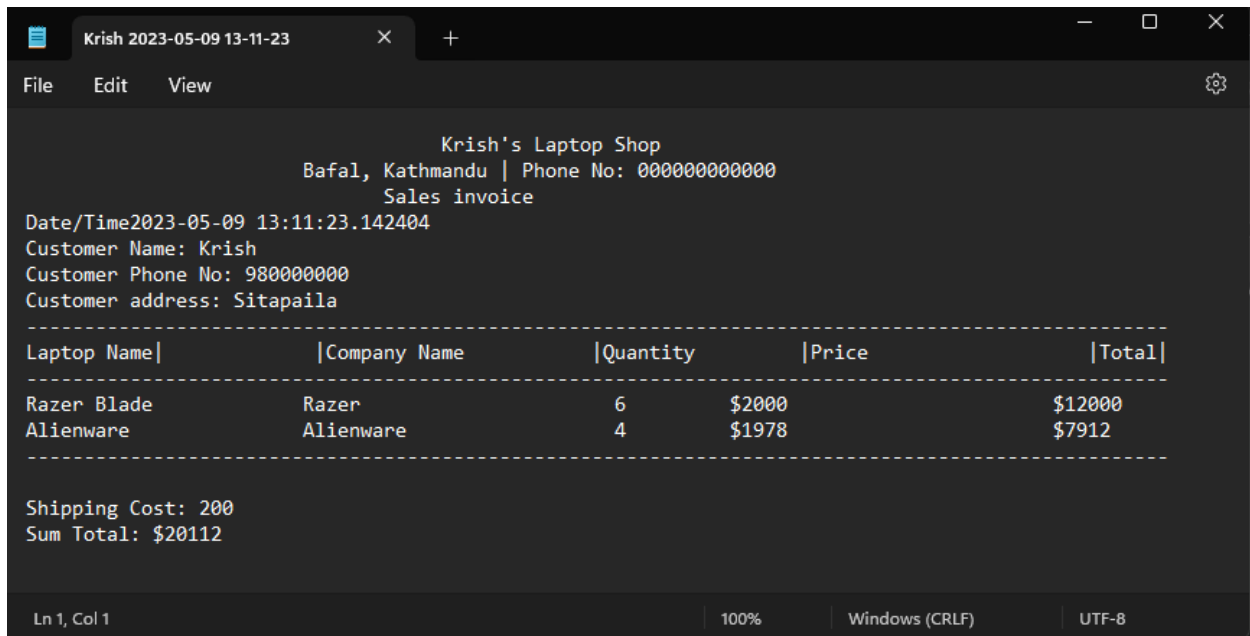


Figure 11: File Generation.



The screenshot shows a text editor window with a dark theme. The title bar indicates the file is named 'Krish 2023-05-09 13-11-23'. The menu bar includes 'File', 'Edit', and 'View'. The main text area contains a sales invoice for 'Krish's Laptop Shop' located in 'Bafal, Kathmandu' with a phone number '00000000000'. The invoice is dated '2023-05-09 13:11:23.142404' and is for customer 'Krish' with phone number '980000000' and address 'Sitapaila'. It lists two items: 'Razer Blade' (6 units at \$2000 each) and 'Alienware' (4 units at \$1978 each). The shipping cost is \$200, and the total sum is \$20112. The status bar at the bottom shows 'Ln 1, Col 1', '100%', 'Windows (CRLF)', and 'UTF-8'.

```
Krish's Laptop Shop
Bafal, Kathmandu | Phone No: 00000000000
Sales invoice
Date/Time2023-05-09 13:11:23.142404
Customer Name: Krish
Customer Phone No: 980000000
Customer address: Sitapaila
-----
Laptop Name|      |Company Name      |Quantity  |Price      |Total|
-----
Razer Blade      Razer           6      $2000      $12000
Alienware        Alienware       4      $1978      $7912
-----
Shipping Cost: 200
Sum Total: $20112
```

Figure 12: Writing of bill in text file.

PURCHASING laptops in the program

```

=====
Welcome to Laptop Shop
=====
Bafal, Kathmandu
=====
1 to Sell
2 to Purchase
3 to Exit
What do you want to do?2
Please enter the name of Distributor: Krish Traders
=====
S.N      Laptop Name      Company Name      Price      Quantity      Processor      GPU
=====
1         Razer Blade       Razer             $2000      170           17 7th Gen     GTX 3060
2         XPS               Dell              $1976      81            15 9th Gen     GTX 3070
3         Alienware        Alienware         $1978      30            15 9th Gen     GTX 3070
4         Swift 7           Acer              $900       50            15 9th Gen     GTX 3070
5         Macbook Pro 16    Apple             $3500      60            15 9th Gen     GTX 3070
=====
Provide the serial number of the laptop you want to purchase: 1

Enter the number of laptops you want to purchase: 4
Would you like to buy more? If YES press Y. If NO press any other button.y

```

Figure 13: Buying of Laptops

```

Would you like to buy more? If YES press Y. If NO press any other button.y
=====
S.N      Laptop Name      Company Name      Price      Quantity      Processor      GPU
=====
1         Razer Blade       Razer             $2000      174           17 7th Gen     GTX 3060
2         XPS               Dell              $1976      81            15 9th Gen     GTX 3070
3         Alienware        Alienware         $1978      30            15 9th Gen     GTX 3070
4         Swift 7           Acer              $900       50            15 9th Gen     GTX 3070
5         Macbook Pro 16    Apple             $3500      60            15 9th Gen     GTX 3070
=====
Provide the serial number of the laptop you want to purchase: 2

Enter the number of laptops you want to purchase: 3
Would you like to buy more? If YES press Y. If NO press any other button.n

```

Figure 14: Buying of Laptops

```

Sasto Laptop Store

Hattiban, Lalitpur | Phone No: 9999999999

Sales Receipt

-----

Print Date|Time: 2023-05-09 12:52:55.411548

Customer Name: Krish Traders

Purchased products:

-----
|Laptop Name|      |Company Name|      |Quantity|      |Price|      |Total|
-----
Razer Blade      Razer              4          $2000      $ 8000
XPS              Dell               3          $1976      $ 5928
-----

Sum Total: $13928

VAT amount: 1810.64

Total amount with VAT Amount: 15738.64

1 to Sell
2 to Purchase
3 to Exit
What do you want to do?

```

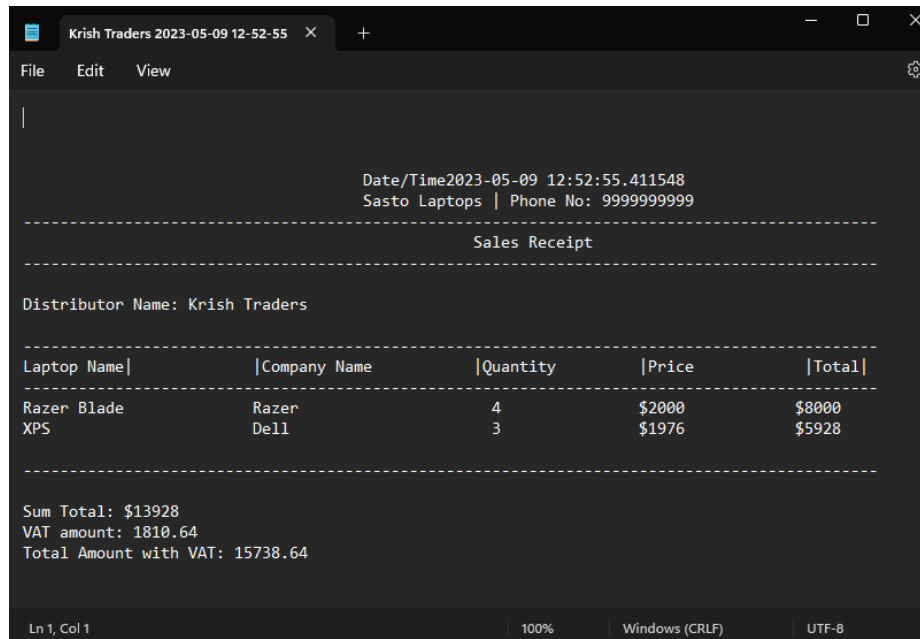
Figure 15: Bill Generation of Purchasing Laptops.

```

Krish Traders 2023-05-09 12-52-55      9/05/2023 12:53 PM      Text Document      1 KB

```

Figure 16: Creation of Text file



```
Krish Traders 2023-05-09 12:52:55 X +
File Edit View
|
Date/Time2023-05-09 12:52:55.411548
Sasto Laptops | Phone No: 9999999999
-----
Sales Receipt
-----
Distributor Name: Krish Traders
-----
Laptop Name|Company Name|Quantity|Price|Total|
-----
Razer Blade    Razer    4    $2000    $8000
XPS            Dell     3    $1976    $5928
-----
Sum Total: $13928
VAT amount: 1810.64
Total Amount with VAT: 15738.64
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Figure 17: Writing of bill in text file.

Termination

```
=====
Welcome to Laptop Shop
=====
Bafal, Kathmandu
=====
1 to Sell
2 to Purchase
3 to Exit
What do you want to do?3
Exit
>>> |
```

Figure 18: Terminating the program.

5. Testing

Test 1 - Implementation of Try, except.

Objective	Test the implementation of Try, Except.
Action	Open IDLE and run the code. Enter invalid inputs to test the implementation of try, except in the code.
Expected Result	An error message should appear, and the code should loop unless the user provides a valid input.
Actual Result	The code loops unless the user provides a valid input.
Conclusion	Test Successful.

Table 1: Testing the implementation of Try, Except.

```

loop = True
while loop == True:
    print("1 to Sell")
    print("2 to Purchase")
    print("3 to Exit")

    while True:
        try:
            user_input = int(input("What do you want to do?"))
            if user_input <=0 or user_input > 3:
                raise ValueError("Please provide a valid number")
            break
        except ValueError:
            print("please provide a valid number")

```

Figure 19: Testing the Implementation of Try, Except

```

=====
Welcome to Laptop Shop
=====
Bafal, Kathmandu
=====
1 to Sell
2 to Buy
3 to Exit
What do you want to do?r
please provide a valid number
What do you want to do?
please provide a valid number
What do you want to do?4
please provide a valid number
What do you want to do?1
Please enter your name:

```

Figure 20: Testing the Implementation of Try, Except

Test 2 – Select purchase and provide negative value as input.

Objective	Provide negative value as input.
Action	Open IDLE and run the code. Enter negative value as input.
Expected Result	An error message should appear, and the code should loop unless the user provides a valid input.
Actual Result	An error message appears and the code loops unless the user provides a valid input.
Conclusion	Test Successful.

Table 2: Providing negative value as input.

```

Please enter the name of Distributor: Harry
-----
S.N      Laptop Name      Company Name      Price      Quantity      Processor      GPU
-----
1         Razer Blade        Razer            $2000      200           17 7th Gen     GTX 3060
2         XPS                Dell             $1976      99            15 9th Gen     GTX 3070
3         Alienware          Alienware        $1978      20            15 9th Gen     GTX 3070
4         Swift 7            Acer             $900       48            15 9th Gen     GTX 3070
5         Macbook Pro 16     Apple            $3500      54            15 9th Gen     GTX 3070
-----
Provide the serial number of the laptop you want to purchase: -1

Please enter a valid serial number:

```

Figure 21: Providing negative value as input.

Test 3-Select the non-existent value as input.

Objective	Provide non- existent value as input.
Action	Open IDLE and run the code. Enter non-existent value as input.
Expected Result	An error message should appear, and the code should loop unless the user provides a valid input.
Actual Result	An error message appears and the code loops unless the user provides a valid input.
Conclusion	Test Successful.

Table 3: Provide non- existent value as input.


```
Please enter the name of Distributor: Harry
-----
S.N      Laptop Name      Company Name      Price      Quantity      Processor      GPU
-----
1        Razer Blade      Razer            $2000      200           17 7th Gen     GTX 3060
2        XPS              Dell             $1976      99            15 9th Gen     GTX 3070
3        Alienware       Alienware       $1978      20            15 9th Gen     GTX 3070
4        Swift 7          Acer            $900       48            15 9th Gen     GTX 3070
5        Macbook Pro 16   Apple           $3500      54            15 9th Gen     GTX 3070
-----
Provide the serial number of the laptop you want to purchase: 6

Please select an available Serial N.O.

Provide the serial number of the laptop you want to purchase:
```

Figure 22: Providing non- existent value as input.

Test 4 – Test the complete purchase process and show the purchased laptops in a text file.

Objective	Test the complete purchase process.
Action	Open IDLE and run the code. Purchasing multiple laptops.
Expected Result	There should be no error while purchasing the laptops and the bill must be generated in a txt file.
Actual Result	There should be no error while purchasing the laptops and the bill must be generated in a txt file.
Conclusion	Test Successful.

Table 4: Testing the complete purchase process.

```

=====
Welcome to Laptop Shop
=====
Bafal, Kathmandu
=====
1 to Sell
2 to Purchase
3 to Exit
What do you want to do?2
Please enter the name of Distributor: Raju
=====
S.N      Laptop Name      Company Name      Price      Quantity      Processor      GPU
-----
1         Razer Blade      Razer             $2000      220           17 7th Gen     GTX 3060
2         XPS              Dell              $1976      105           15 9th Gen     GTX 3070
3         Alienware       Alienware         $1978      23            15 9th Gen     GTX 3070
4         Swift 7          Acer              $900       48            15 9th Gen     GTX 3070
5         Macbook Pro 16   Apple             $3500      54            15 9th Gen     GTX 3070
=====
Provide the serial number of the laptop you want to purchase: 3

Enter the number of laptops you want to purchase: 7
Would you like to buy more? If YES press Y. If NO press any other button.y

```

Figure 23: Testing the complete purchase process.

Would you like to buy more? If YES press Y. If NO press any other button.y

S.N	Laptop Name	Company Name	Price	Quantity	Processor	GPU
1	Razer Blade	Razer	\$2000	220	17 7th Gen	GTX 3060
2	XPS	Dell	\$1976	105	15 9th Gen	GTX 3070
3	Alienware	Alienware	\$1978	30	15 9th Gen	GTX 3070
4	Swift 7	Acer	\$900	48	15 9th Gen	GTX 3070
5	Macbook Pro 16	Apple	\$3500	54	15 9th Gen	GTX 3070

Provide the serial number of the laptop you want to purchase: 5

Enter the number of laptops you want to purchase: 6
Would you like to buy more? If YES press Y. If NO press any other button.n

Figure 24: Testing the complete purchase process.

Sasto Laptop Store

Hattiban, Lalitpur | Phone No: 9999999999

Sales Receipt

Print Date|Time: 2023-05-08 14:27:49.374533

Customer Name: Raju

Purchased products:

Laptop Name	Company Name	Quantity	Price	Total
Alienware	Alienware	7	\$1978	\$ 13846
Macbook Pro 16	Apple	6	\$3500	\$ 21000

Sum Total: \$34846

VAT amount: 4529.9800000000005

Total VAT Amount: 39375.98

Figure 25: Testing the complete purchase process.

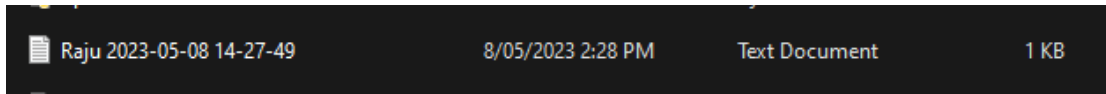


Figure 26: Testing the complete purchase process.

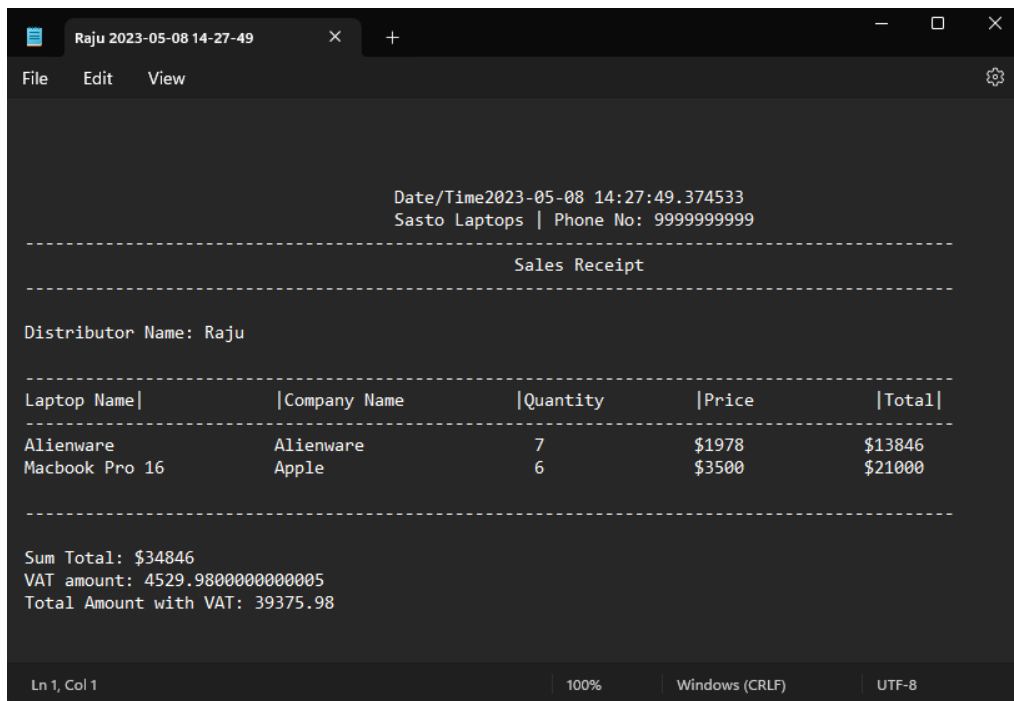


Figure 27: Testing the complete purchase process.

Test 5 – Test the complete sell process and show the sold laptops in a text file.

Objective	Test the selling process.
Action	Open IDLE and run the code. Selling multiple laptops.
Expected Result	There should be no error while selling the laptops and the bill must be generated in a text file.
Actual Result	There was no error while selling the laptops and the bill must be generated in a txt file.
Conclusion	Test Successful.

Table 5: Testing the selling process.

```

=====
Welcome to Laptop Shop
=====
Bafal, Kathmandu
=====
1 to Sell
2 to Purchase
3 to Exit
What do you want to do?1
Please enter your name: Krish

Please enter your phone no: 9800000000

Please enter your Address: Sitapaila
=====
S.N      Laptop Name      Company Name      Price      Quantity      Processor      GPU
=====
1         Razer Blade      Razer             $2000      175           17 7th Gen     GTX 3060
2         XPS              Dell              $1976      85            15 9th Gen     GTX 3070
3         Alienware        Alienware         $1978      30            15 9th Gen     GTX 3070
4         Swift 7          Acer              $900       48            15 9th Gen     GTX 3070
5         Macbook Pro 16   Apple             $3500      60            15 9th Gen     GTX 3070
=====
Provide the serial number of the laptop you want to purchase: 1

Enter the number of laptops you want to purchase: 5
Would you like to purchase more? If yes press Y. If no press any other button.y

```

Figure 28: Test the selling process.

```

Would you like to purchase more? If yes press Y. If no press any other button.y

=====
S.N      Laptop Name      Company Name      Price      Quantity      Processor      GPU
=====
1         Razer Blade      Razer             $2000      170           17 7th Gen     GTX 3060
2         XPS              Dell              $1976      85            15 9th Gen     GTX 3070
3         Alienware        Alienware         $1978      30            15 9th Gen     GTX 3070
4         Swift 7          Acer              $900       48            15 9th Gen     GTX 3070
5         Macbook Pro 16   Apple             $3500      60            15 9th Gen     GTX 3070
=====
Provide the serial number of the laptop you want to purchase: 2

Enter the number of laptops you want to purchase: 5
Would you like to purchase more? If yes press Y. If no press any other button.n

```

Figure 29: Testing the selling process.

```

                                Krish's Laptop Shop

                                Bafal, Kathmandu | Phone No: 0000000000000

                                Sales invoice

Date|Time: 2023-05-08 15:12:07.334581
Customer Name: Krish
Customer Phone No: 9800000000
Customer Address: Sitapaila

-----
Laptop Name|                |Company Name                |Quantity|Price|                |Total|
-----
Razer Blade                Razer                        5        $2000                $ 10000
XPS                        Dell                        5        $1976                $ 9880
-----
Shipping Cost: 200
Sum Total: $20080

1 to Sell
2 to Purchase
3 to Exit
What do you want to do?
```

Figure 30: Test the selling process.

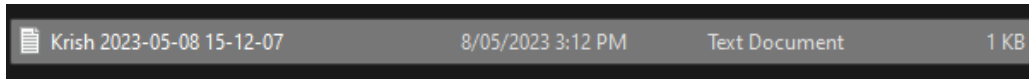


Figure 31: Testing the selling process.

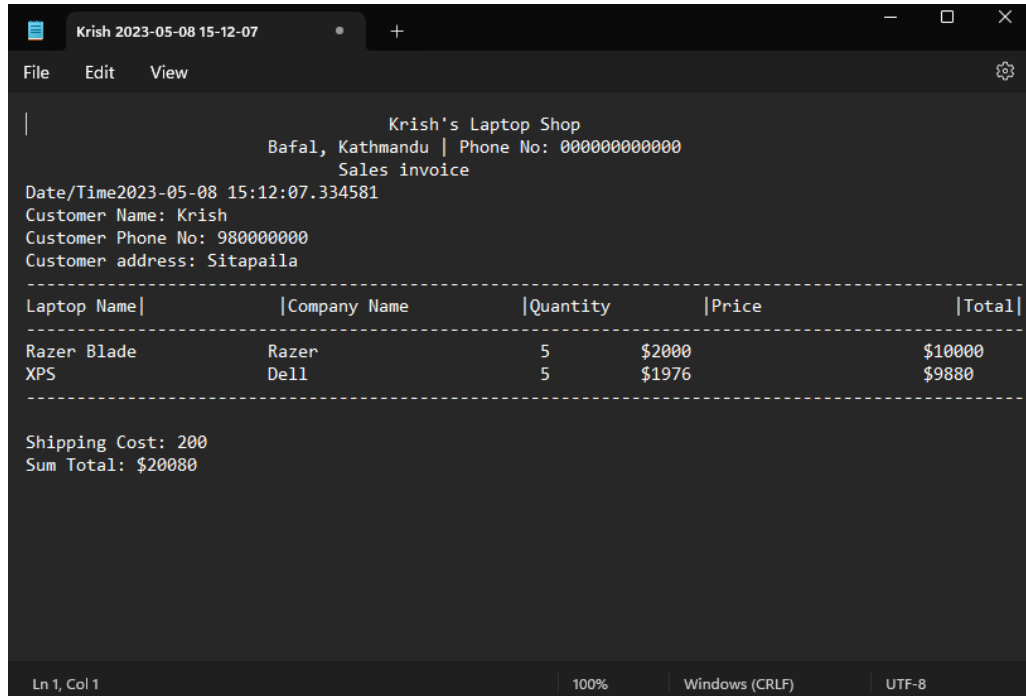
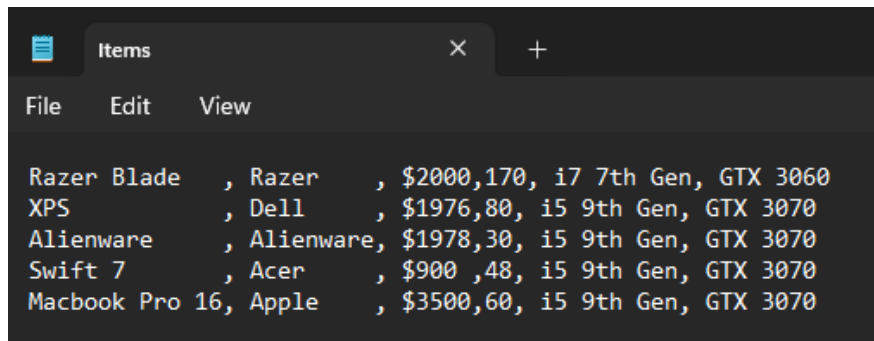
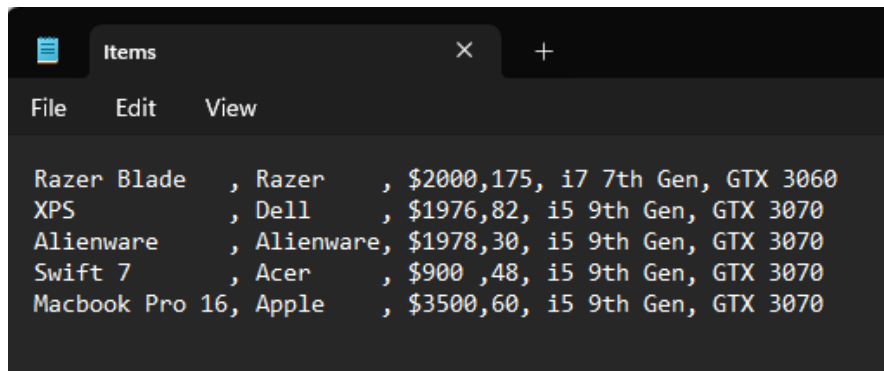


Figure 32: Testing the selling process.

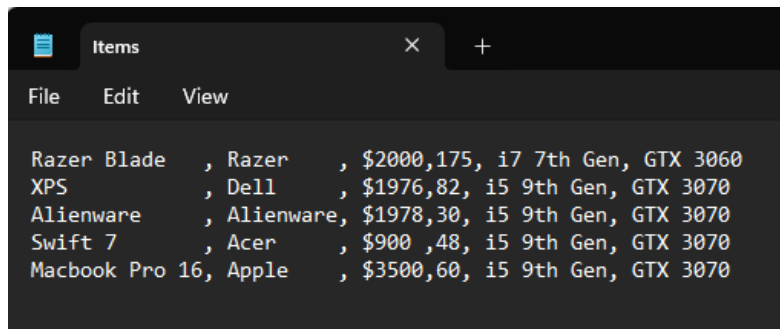
Test 6 – Test the quantity being added while purchasing the laptop.

Objective	Test the quantity being added while purchasing the laptop.
Action	Open the “Items.txt” file. Inspect the “Items.txt” file.
Expected Result	The quantity should be added.
Actual Result	The quantity is added.
Conclusion	Test Successful.

Table 6: Testing the quantity being added while purchasing the laptop.*Figure 33: Testing the quantity being added while purchasing the laptop.**Figure 34: Testing the quantity being added while purchasing the laptop.*

Test 7 – Test the quantity being deducted while selling the laptop.

Objective	Test the quantity being deducted while selling the laptop.
Action	Open the “Items.txt” file. Inspect the “Items.txt” file.
Expected Result	The quantity should be deducted
Actual Result	The quantity was deducted.
Conclusion	Test Successful.

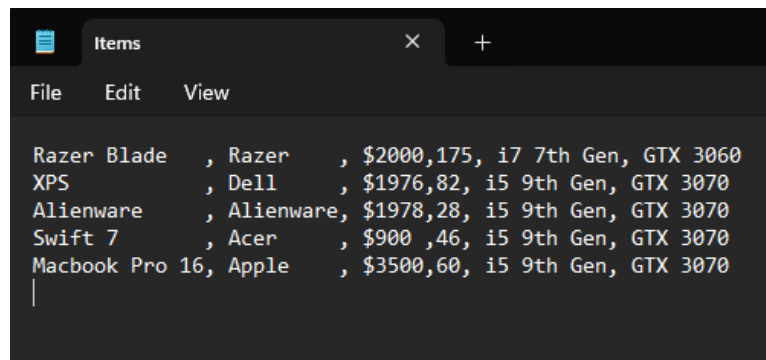
Table 7: Testing the quantity being deducted while selling the laptop.


```

Items
File Edit View

Razer Blade , Razer , $2000,175, i7 7th Gen, GTX 3060
XPS , Dell , $1976,82, i5 9th Gen, GTX 3070
Alienware , Alienware, $1978,30, i5 9th Gen, GTX 3070
Swift 7 , Acer , $900 ,48, i5 9th Gen, GTX 3070
Macbook Pro 16, Apple , $3500,60, i5 9th Gen, GTX 3070

```

Figure 35: Testing the quantity being deducted while selling the laptop.


```

Items
File Edit View

Razer Blade , Razer , $2000,175, i7 7th Gen, GTX 3060
XPS , Dell , $1976,82, i5 9th Gen, GTX 3070
Alienware , Alienware, $1978,28, i5 9th Gen, GTX 3070
Swift 7 , Acer , $900 ,46, i5 9th Gen, GTX 3070
Macbook Pro 16, Apple , $3500,60, i5 9th Gen, GTX 3070
|

```

Figure 36: Testing the quantity being deducted while selling the laptop.

6. Conclusion

In conclusion, this program controls the sales and inventory of a laptop rental business. The software is made to read a text file named 'Items.txt' which provides the details about the laptops that are available, display them to customers, and handle stock and transactions. The application stores and manipulates the data using data structures including dictionaries, lists, and strings.

The program's implementation is modular, with distinct functions for input/output, reading files, producing the invoice. The application validates inputs from the customers.

The report includes diagrams, flowcharts, and other visual aids to describe the program's behavior and structure in depth.

Bibliography

DataCamp, 2023. *Data Structures in Python*. [Online]
Available at: <https://www.datacamp.com/tutorial/data-structures-python>
[Accessed 10 5 2023].

Dataquest, 2023. *Data Structures in Python*. [Online]
Available at: <https://www.dataquest.io/blog/data-structures-in-python/>
[Accessed 10 5 2023].

GeeksforGeeks, 2023. *How to write a Pseudo Code?*. [Online]
Available at: <https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/>
[Accessed 10 5 2023].

GeeksforGeeks, 2023. *What is a Flowchart and its Types?*. [Online]
Available at: <https://www.geeksforgeeks.org/what-is-a-flowchart-and-its-types/>
[Accessed 9 5 2023].

PythonGeeks, 2023. *Python Data Structures - A Complete Overview*. [Online]
Available at: <https://pythongeeks.org/python-data-structures/>
[Accessed 10 5 2023].

Rossum, G. v., 2023. *Python: Design and History*. [Online]
Available at: <https://www.python.org/doc/essays/blurb/>
[Accessed 9 5 2023].

Simplilearn, 2023. *What is an Algorithm? Definition, Examples, and Analysis*. [Online]
Available at: <https://www.simplilearn.com/tutorials/data-structure-tutorial/what-is-an-algorithm>
[Accessed 10 5 2023].

7. Appendix

Code for Main.py

```
from read import *
from write import *
from operations import *
import datetime
now = datetime.datetime.now()
date_time = now.strftime("%Y-%m-%d %H-%M-%S")
# import operations

print("\n")
print("=====")
print("\t Welcome to Laptop Shop")
print("=====")
print("\t Bafal, Kathmandu")
print("=====")

laptop_dictionary = laptops()

# Asking and validating what the user wants to do

loop = True
while loop == True:
    print("1 to Sell")
```

```
print("2 to Purchase")
```

```
print("3 to Exit")
```

```
while True:
```

```
    try:
```

```
        user_input = int(input("What do you want to do?"))
```

```
        if user_input <=0 or user_input > 3:
```

```
            raise ValueError("Please provide a valid number")
```

```
        break
```

```
    except ValueError:
```

```
        print("please provide a valid number")
```

```
if user_input == 1:
```

```
    user_name, phone_number, user_address = sell()
```

```
    purchased_laptops = plaptop()
```

```
buy_more = True
```

```
while buy_more == True:
```

```
    table1()
```

```
    # Calling the function from read.py
```

```
    items()
```

#Calling the function from operation.py

serial_no = serial()

#Valid the serial number

while serial_no <= 0 or serial_no > len(laptop_dictionary):

#Calling the function from operation.py

serial_no = checkserial()

#Valid Quantity

selected_laptop_quantity = valid(laptop_dictionary, serial_no)

#Valid Quantity

user_quantity = change_quantity(selected_laptop_quantity)

#Valid Quantity

#Update the text file

update(laptop_dictionary, serial_no, user_quantity)

#Update the text file

#getting user purchased items

product_name, unit_price, total_price, company_name = get(laptop_dictionary,
serial_no, user_quantity, purchased_laptops)

shipping_cost, user_request = more()

```
#Validating the user input

if user_request == "Y":

    buy_more = True

else:

    total = 0

    for i in purchased_laptops:

        total+=int(i[4])

    sum_total = total + shipping_cost


# calling the function from write.py

    printsell(now, user_name, phone_number, user_address, shipping_cost,
sum_total, purchased_laptops)


# calling the function from write.py

    sellbill(user_name, date_time, now, phone_number, user_address,
purchased_laptops, shipping_cost, sum_total)


    buy_more = False


elif user_input == 2:

    #Getting the Distributor Name

    while True:

        distributor_name = input("Please enter the name of Distributor: ")
```



```
    if distributor_name == "":
        print("Textfield is empty.")
    else:
        break

purchased_laptops = plaptop()

buy_more = True
while buy_more == True:
    table2()

    #calling the function from read.py
    items()

    serial_no = serial2()

    #Validate the provided serial number
    while serial_no <= 0 or serial_no > len(laptop_dictionary):
        serial_no = checkserial()

    #Valid Quantity
    selected_laptop_quantity, user_quantity = valid2(laptop_dictionary, serial_no)
    #Valid Quantity

    #Update the text file
```

```
update2 (laptop_dictionary, serial_no, user_quantity)

#Update the text file

#getting user purchased items

    product_name, unit_price, total_price, item_price, company_name =
get2(laptop_dictionary, serial_no, user_quantity, purchased_laptops)

user_request = more2()

# Loop the code if the user inputs y

if user_request == "Y":

    buy_more = True

else:

    # Print bill if the user inputs any other value except y

    buy_more = False

printbuy(now, distributor_name, purchased_laptops)

buybill(distributor_name, date_time, now, purchased_laptops)

# If the user inputs '3' terminate the program

elif user_input == 3:

    loop = False
```

```
print ("Exit")
```

Code for Operations.py

```
import datetime
```

```
from write import *
```

```
def sell():
```

```
    # Get the user information and validate it
```

```
    while True:
```

```
        user_name = input("Please enter your name: ")
```

```
        if user_name == "":
```

```
            print("Empty Textfield.")
```

```
        else:
```

```
            break
```

```
    print("\n")
```

```
    while True:
```

```
        phone_number = (input("Please enter your phone no: "))
```

```
        if phone_number.isdigit():
```

```
            phone_number = int(phone_number)
```

```
            break
```

```
        elif phone_number == "":
```

```
            print("Please enter a valid phone number!!!")
```

```
        else:
```

```
            print("Please enter a valid phone number!!!")
```

```
print("\n")

while True:

    user_address = input("Please enter your Address: ")

    if user_address == "":

        print("Empty Textfield.")

    else:

        break

return user_name, phone_number, user_address
```

```
def plaptop():

    purchased_laptops = []

    return purchased_laptops
```

```
def table1():

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

    print("S.N \t\t Laptop Name \t\t Company Name \t\t Price \t\t Quantity \t Processor \t\t GPU")

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

#Ask the user for the serial number

```
def serial():
```

```
    serial_no = (input("Provide the serial number of the laptop you want to purchase: "))
```

```
    print("\n")
```

```
    while not serial_no.isdigit():
```

```
        serial_no = input("Please enter a valid serial number: ")
```

```
    serial_no = int(serial_no)
```

```
    return serial_no
```

#Ask the user for the serial number if the user inputs an invalid serial number

```
def checkserial():
```

```
    # while serial_no <= 0 or serial_no > len (laptop_dictionary):
```

```
    print("Please select an available Serial N.O.")
```

```
    print("\n")
```

```
    serial_no = int(input("Provide the serial number of the laptop you want to purchase: "))
```

```
    print("\n")
```

```
    return serial_no
```

#Valid the Quantity

```
def valid(laptop_dictionary, serial_no):
```

```
    selected_laptop_quantity = laptop_dictionary[serial_no][3]
```

```
    return selected_laptop_quantity
```

```
def change_quantity(selected_laptop_quantity):
```

```
    while True:
```

```
        try:
```

```
            user_quantity = input("Enter the number of laptops you want to purchase: ")
```

```
            if not user_quantity:
```

```
                raise ValueError("Please provide a value")
```

```
            user_quantity = int(user_quantity)
```

```
            if user_quantity <= 0 or user_quantity > int(selected_laptop_quantity):
```

```
                raise ValueError("The quantity you are looking for is unavailable to purchase.")
```

```
            return user_quantity
```

```
        except ValueError:
```

```
            print("Invalid input: ")
```

```
#Update the text file
```

```
def update(laptop_dictionary, serial_no, user_quantity):
```

```
    laptop_dictionary[serial_no][3] = int(laptop_dictionary[serial_no][3]) - int(user_quantity)
```

```
    dictionary(laptop_dictionary)
```

```
#get the purchased items from the user
```

```
def get(laptop_dictionary, serial_no, user_quantity, purchased_laptops):
```

```
product_name = laptop_dictionary[serial_no][0]
selected_laptop_quantity = user_quantity
unit_price = laptop_dictionary[serial_no][2]
item_price = laptop_dictionary[serial_no][2].replace("$", "")
total_price = int(item_price)*int(selected_laptop_quantity)
company_name = laptop_dictionary[serial_no][1]

        purchased_laptops.append([product_name,          company_name,
selected_laptop_quantity, unit_price, total_price])

    return product_name, unit_price, total_price, company_name


#Ask if the user wants to Buy more

def more():

    user_request = input("Would you like to purchase more? If yes press Y. If no press any
other button.").upper()

    print("\n")

    shipping_cost = 200

    return shipping_cost, user_request


#Purchase code

def table2 ():
```

```
print("-----")
print("S.N \t\t Laptop Name \t\t Company Name \t\t Price \t\t Quantity \t Processor \t\t GPU")
print("-----")
```

```
def serial2 ():
```

```
    #Get the provided serial number and validate it
```

```
    serial_no = (input("Provide the serial number of the laptop you want to purchase: "))
```

```
    print("\n")
```

```
    while not serial_no.isdigit():
```

```
        serial_no = input("Please enter a valid serial number: ")
```

```
    serial_no = int(serial_no)
```

```
    return serial_no
```

```
    #Valid Quantity
```

```
def valid2 (laptop_dictionary, serial_no):
```

```
    while True:
```

```
        user_input = input("Enter the number of laptops you want to purchase: ")
```

```
        if user_input.strip() == "":
```

```
            print("Empty input")
```

```
        elif not user_input.isdigit():
```



```
        print("Invalid input. Please enter a valid number.")

    else:

        user_quantity = int(user_input)

        break

    selected_laptop_quantity = laptop_dictionary[serial_no][3]

    return selected_laptop_quantity, user_quantity

def update2(laptop_dictionary, serial_no, user_quantity):

    laptop_dictionary[serial_no][3] = int(laptop_dictionary[serial_no][3]) + int(user_quantity)

    dictionary (laptop_dictionary)

    #getting user purchased items

def get2(laptop_dictionary, serial_no, user_quantity, purchased_laptops):

    product_name = laptop_dictionary[serial_no][0]

    selected_laptop_quantity = user_quantity

    unit_price = laptop_dictionary[serial_no][2]

    item_price = laptop_dictionary[serial_no][2].replace("$", "")

    total_price = int(item_price)*int(selected_laptop_quantity)

    company_name = laptop_dictionary[serial_no][1]

    purchased_laptops.append([product_name, company_name,
selected_laptop_quantity, unit_price, total_price])

    return product_name, unit_price, total_price, item_price, company_name
```

#getting user purchased items

```
def more2 ():
```

```
    user_request = input("Would you like to buy more? If YES press Y. If NO press any  
    other button.").upper() #Converts value into Upper Case
```

```
    print("\n")
```

```
    return user_request
```

Code for read.py

```
def laptops():
```

```
    file = open ("Items.txt", "r")
```

```
    laptop_dictionary = {}
```

```
    laptop_id = 1
```

```
    for line in file:
```

```
        line = line.replace("\n", "")
```

```
        laptop_dictionary.update({laptop_id: line.split(",")})
```

```
        laptop_id = laptop_id + 1
```

```
    file.close
```

```
    return laptop_dictionary
```

```
def items():
```

```
    file = open("Items.txt", "r")
```

```
    a = 1 #serial no
```

```
    for line in file:
```

```

    print(a, "\t\t"+line.replace(", ", "\t\t"))

    a = a+1 #Serial Number

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

    file.close()

```

Code for Write.py

```

def dictionary(laptop_dictionary):

    file = open("Items.txt", "w")

    for values in laptop_dictionary.values():

        file.write(str(values[0]) + "," +
+str(values[1])+" "+str(values[2])+" "+str(values[3])+" "+str(values[4])+" "+str(values[5]))

        file.write("\n")

    file.close()

```

#SELL BILL (WRITE)

```

def sellbill(user_name, date_time, now, phone_number, user_address,
purchased_laptops, shipping_cost, sum_total):

    filename = f"{user_name} {date_time}.txt"

    file = open(filename, "w")

    file.write("\t\t\t\t\tKrish's Laptop Shop")

    file.write("\n")

    file.write("\t\t\t\t\tBafal, Kathmandu | Phone No: 0000000000000")

    file.write("\n")

    file.write("\t\t\t\t\tSales invoice")

```

```

file.write("\n")

file.write("Date/Time"+str(now),)

file.write("\n")

file.write("Customer Name: "+str(user_name))

file.write("\n")

file.write("Customer Phone No: "+str(phone_number))

file.write("\n")

file.write("Customer address: "+str(user_address))

file.write("\n")

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

file.write("Laptop Name| \t\t |Company Name \t\t |Quantity \t\t |Price \t\t\t " " |Total| "
"\n")

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

for i in purchased_laptops:

    file.write (str(i[0])+"\t " +str(i[1])+" \t\t "+" " +str(i[2])+" \t " +str(i[3]) + "\t\t\t "
+ "$"+ str(i[4]) + "\n")

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

file.write("\n")

file.write("Shipping Cost: "+ str(shipping_cost))

file.write("\n")

file.write("Sum Total: $"+str(sum_total))

file.write("\n")

```



```

file.write("\n")

file.write("\n")

file.write("Distributor Name: " + str(distributor_name))

file.write("\n")

file.write("\n")

file.write("-----")

file.write("\n")

file.write("Laptop Name| \t\t |Company Name \t\t |Quantity \t\t |Price \t\t |Total| ")

file.write("\n")

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

# for i in purchased_laptops:

# file.write (str(i[0])+"\t " +str([1])+ "\t\t " +str(i[2])+ "\t\t " + str(i[3]) +"\t " +
"$" + str([4]) + "\n")

for i in purchased_laptops:

    file.write (str(i[0])+"\t " " " +str(i[1])+ " \t\t " + " " +str(i[2])+ "\t\t " +str(i[3]) +
"\t " +"$" +str(i[4]) + "\n")

file.write("\n")

file.write("-----")

file.write("\n")

file.write("\n")

file.write("Sum Total: $" +str(sum_total))

file.write("\n")

file.write("VAT amount: " +str(vat))

file.write("\n")

file.write("Total Amount with VAT: " +str(vat_amt))

```

#SELL BILL

```
def printsell(now, user_name, phone_number, user_address, shipping_cost, sum_total,
purchased_laptops):
```

```
    print("\n")
```

```
    print("\t\t\t\t\tKrish's Laptop Shop")
```

```
    print("\n")
```

```
    print("\t\t\t\tBafal, Kathmandu | Phone No: 00000000000000")
```

```
    print("\n")
```

```
    print("\t\t\t\t\tSales invoice")
```

```
    print("\n")
```

```
    print("Date|Time: "+str(now),)
```

```
    print("Customer Name: "+str(user_name))
```

```
    print("Customer Phone No: "+str(phone_number))
```

```
    print("Customer Address: "+str(user_address))
```

```
    print("\n")
```

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

```
    print("Laptop Name| \t\t\t |Company Name \t\t\t |Quantity \t\t |Price \t\t\t |Total| ")
```

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

```
    print("\n")
```

```
    for i in purchased_laptops:
```

```
        print (i[0], "\t\t\t", i[1], "\t\t\t", "    ", i[2], "\t\t", i[3], "\t\t\t", "$", i[4], "\n")#, i[4], "\t\t\t", i[5])
```

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

print("Shipping Cost: "+ str(shipping_cost))

print("Sum Total: $" +str(sum_total))

print("\n")

print("\n")

print("\n")

#BUY BILL

def printbuy(now, distributor_name, purchased_laptops):

    total = 0

    for i in purchased_laptops:

        total+=int(i[4])

    sum_total = total

    vat = sum_total * (13/100)

    vat_amt = sum_total + vat

    print("\n")

    print("\t\t\t\t\tSasto Laptop Store")

    print("\n")

    print("\t\t\t\t\tHattiban, Lalitpur | Phone No: 9999999999")

    print("\n")

    print("\t\t\t\t\tSales Receipt")

    print("\n")

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



```
print("\n")

print("Print Date|Time: "+str(now),)

print("\n")

print("Customer Name: "+str(distributor_name))

print("\n")

print("Purchased products: ")

print("\n")

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

print("Laptop Name| \t\t\t |Company Name \t\t\t " " |Quantity \t\t |Price \t\t\t |Total| ")

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

print("\n")

for i in purchased_laptops:

    print (i[0], "\t\t\t", i[1], "\t\t\t", "    ", i[2], "\t\t", i[3], "\t\t\t", "$", i[4], "\n")#, i[4], "\t\t\t", i[5])

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

print("\n")

print("Sum Total: $"+str(sum_total))

print("\n")

print("VAT amount: "+str(vat))

print("\n")

print("Total amount with VAT Amount: "+str(vat_amt))

print("\n")
```