LONDON METROPOLITAN UNIVERSITY

ITAHARI INTERNATIONAL COLLEGE

Module Code & Module Title:

CS4001NT Programming

Assessment Weightage & Type:

60% Individual Coursework

Year and Semester:

2022 Autumn

Student Name: Aayush Wanem Limbu

London Met ID: 22072043

College ID:np05cp4a220010

Assignment Due Date:

Word Count: 7451

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

Contents

Table of Figure

Table of Table

1. Introduction

This coursework is of module "**Fundamentals of Computing"** which carries 60% of total module. It is an individual coursework where students must create a management system for a laptop rental store using python as a programming language. The primary objective of this coursework is to create a functional and efficient management system specifically designed for a laptop rental store. Through the implementation of Python programming, students will have the opportunity to apply their knowledge of fundamental computing concepts and techniques to real-world scenarios. By developing the management system, students will gain practical experience in utilizing Python's features and functionalities, such as data handling and algorithms.

Brief Introduction About the Project

On this project, we were asked to create a management system for a laptop rental store. We were given a scenario where a laptop shop buys laptops and computers from the manufacturers and sell it to various customers which may be individuals or companies. Also, Customers can place orders for laptop as well. The system functions for two user roles, admin, and customer.

Admin can add stocks to the system and can see the purchase details of products. The system will maintain the stocks for admin to keep track of stocks. Similarly, when a customer login into the system, it displays all the products available in the store. When a customer bus any product an invoice is generated on the same name which they have used while logging in. Also, a customer can buy another product in the same login session.

The steps and developing process are thoroughly explained in the following sections.

1.1 **goals and objectives**

Aims

This project aims to create a management system for a laptop rental store. The management system will streamline the operations of the store, from purchasing laptops from manufacturers to selling them out to customers.

Objectives:

The objectives of a shop management system can vary depending on the specific needs of the business, but some common ones may include:

1. **Streamlining operations:** The system should aim to automate and simplify various processes involved in running the shop, such as inventory management, order processing, and sales tracking. This can help to reduce manual errors and improve overall efficiency.

2. **Enhancing customer experience:** The system should make it easier for customers to shop, browse products, and make purchases. This can include features like online ordering, personalized recommendations, and easy checkout options.

3. **Improving inventory management:** The system should provide real-time inventory tracking and help to optimize stock levels. This can reduce waste and improve overall profitability.

4. **Boosting sales and revenue:** The system should help to identify sales trends and customer preferences, allowing the business to make informed decisions about product offerings and marketing strategies.

5. **Enhancing reporting and analytics:** The system should provide detailed insights into various aspects of the business, such as sales performance, customer behavior, and inventory levels. This can help the business to make data-driven decisions and improve overall performance.

2 Discussion and analysis

2.1 Algorithm:

An algorithm is a procedure used for solving a problem or performing a computation. Algorithms act as an exact list of instructions that conduct specified actions step by step in either hardware- or software-based routines. Algorithms are widely used throughout all areas of IT. In mathematics and computer science, an algorithm usually refers to a small procedure that solves a recurrent problem. Algorithms are also used as specifications for performing data processing and play a major role in automated systems. An algorithm could be used for sorting sets of numbers or for more complicated tasks, like recommending user content on social media. Algorithms typically start with initial input and instructions that describe a specific computation. When the computation is executed, the process produces an output. (Gills, 2022)

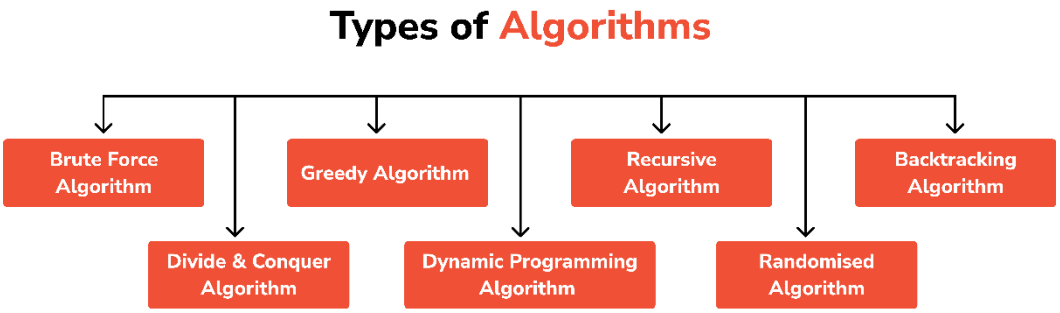


Figure 1 Types of Algorithms.

2.1.1 Program as a one whole Algorithm

This program is based on the concept of a main function and many sub functions which are called when the appropriate action the function does is triggered or called. The program as a one whole algorithm is given below:

Step1: Start

Step 2. Import necessary libraries: datetime, tabulate, getpass, termcolor

Step 3. Define the function "customer_buy":

3.1. Read the contents of the "Stocks.txt" file and store them in "existing_stocks" variable.

3.2. Print the list of available laptops.

3.3. Create an empty list called "laptops".

3.4. Iterate over each stock item in "existing_stocks" and extract the laptop name, appending it to the "laptops" list.

3.5. Call the "stocks" function to display the formatted stocks.

3.6. Prompt the user to enter the name of the laptop they want to buy until a valid laptop name is entered.

3.7. Prompt the user to enter the quantity they want to buy until a valid quantity is entered.

3.8. Iterate over each stock item in "existing_stocks":

8.1. Extract the laptop details from the stock item.

8.2. If the laptop name matches the selected laptop:

8.2.1. Extract the current stock quantity.

8.2.2. If the current stock quantity is less than the selected quantity, display an error message and return.

3.8.2.3. Calculate the new stock quantity.

3.8.2.4. Update the stock item with the new quantity.

3.8.2.5. Display a success message and generate the invoice.

3.8.2.6. Break the loop.

3.9. Write the updated file contents back to the "Stocks.txt" file.

3.10. Prompt the user to enter their name.

3.11. Open a new file with the customer's name and write the invoice details:

3.11.1. Get the current date and time.

3.11.2. Calculate the price per unit.

3.11.3. Calculate the VAT amount.

3.11.4. Calculate the shipping cost.

3.11.5. Calculate the total price without VAT.

3.11.6. Calculate the final amount including VAT and shipping cost.

3.11.7. Write the formatted invoice details to the file.

3.12. End of the "customer_buy" function.

Step 4. Define the function "customer_login":

4.1. Display the options available to the customer.

4.2. Prompt the user to enter their choice until a valid option is selected.

4.3. Based on the selected option:

4.3.1. If the option is 1, call the "stocks" function to display the available stocks.

4.3.2. If the option is 2, call the "customer_buy" function.

4.3.3. If the option is 3, break the loop and return to the previous state.

4.3.4. If the option is 4, prompt for confirmation to exit.

4.3.4.1. If confirmed, exit the program.

4.3.4.2. If not confirmed, continue.

4.3.5. If none of the valid options are selected, display an error message.

4.4. End of the "customer_login"

Step 5. Start the main program:

Step 6. Call the "customer_login" function to initiate the customer login process.

Step 7. Inside the "customer_login" function:

7.1. Display the options available to the customer.

7.2. Prompt the user to enter their choice until a valid option is selected.

7.3. Based on the selected option:

7.3.1. If the option is 1, call the "stocks" function to display the available stocks.

7.3.2. If the option is 2, call the "customer_buy" function.

7.3.3. If the option is 3, break the loop and return to the previous state.

7.3.4. If the option is 4, prompt for confirmation to exit.

7.3.4.1. If confirmed, exit the program.

7.3.4.2. If not confirmed, continue.

7.3.5. If none of the valid options are selected, display an error message.

7.4. Repeat steps 7.1 to 7.3 until the user chooses to exit.

Step 8. End of the main program.

2.2 Pseudo code:

Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.

The rules of Pseudocode are reasonably straightforward. All statements showing "dependency" are to be indented. These include while, do, for, if, switch. Examples below will illustrate this notion.

Examples:

1.. If student's grade is greater than or equal to 60

Print "passed"

else

Print "failed"

2. Set total to zero

Set grade counter to one

While grade counter is less than or equal to ten

Input the next grade

Add the grade into the total

Set the class average to the total divided by ten

Print the class average.

(University of North Florida, 2023/05/10)

2.2.1 Pseudo code of program

Given below is the pseudo code of the program:

```
FUNCTION customer_buy()
   OPEN "Stocks.txt" FOR READING AS stockks_file
   SET existing_stocks TO READLINES(stocks_file)
   CLOSE stocks_file

   OUTPUT ""
   OUTPUT "------------------"
   OUTPUT "Available laptops:"

   SET laptops TO []
   FOR EACH stock_item IN existing_stocks DO
      SET item_parts TO SPLIT(stock_item, ", ")
      APPEND item_parts[0] TO laptops
   ENDFOR

   CALL stocks()
   OUTPUT ""
   WHILE True DO
      SET selected_laptop TO INPUT "Enter the name of the laptop you want to buy:"
      IF selected_laptop NOT IN laptops THEN
         OUTPUT "Invalid laptop name. Please try again."
      ELSE
         BREAK LOOP
      ENDIF
   ENDWHILE

   WHILE True DO
      SET selected_quantity TO INPUT "Enter the quantity you want to buy: "
      TRY
         IF NOT ISDIGIT(selected_quantity) THEN
            RAISE ERROR "Invalid quantity! Please enter a number."
         ELSEIF INTEGER(selected_quantity) <= 0 THEN
            RAISE ERROR "Quantity must be greater than zero."
         ENDIF

         BREAK LOOP
```

```
        CATCH ERROR AS e DO
            OUTPUT e
        ENDCATCH
    ENDWHILE
    FOR i FROM 0 TO LENGTH(existing_sTOcks)-1 DO
        SET item_parts TO SPLIT(existing_sTOcks[i], ", ")
        IF item_parts[0] == selected_laptop THEN
            SET current_stock TO INT(item_parts[3])

            IF current_stock < INT(selected_quantity) THEN
                OUTPUT f"Sorry, we only have {current_stock} {selected_laptop}(s) in
stock."
                RETURN
            ENDIF

            SET new_stock TO current_stock - INT(selected_quantity)
            existing_stocks[i] = f"{item_parts[0]}, {item_parts[1]}, {item_parts[2]},
{new_sTOck}, {item_parts[4]}, {item_parts[5]}\n"

            OUTPUT f"Purchase of {selected_quantity} {selected_laptop}(s) was
successful!"
        ENDIF
    ENDFOR
    OPEN "Stocks.txt" FOR WRITING AS stocks_file
    WRITE existing_stocks TO stocks_file
    CLOSE stocks_file

ENDFUNCTION

// CALL the customer_buy FUNCTION TO start the program
customer_buy()

FUNCTION customer_login():
    WHILE True DO
        OUTPUT ""
        OUTPUT "Dear customer, welcome to our shop."
        OUTPUT "What would you like to do today?"
        OUTPUT "Please choose from the options below:"
        OUTPUT ""
        OUTPUT "1. Show available pieces in stock"
        OUTPUT "2. Buy laptops"
```

```
OUTPUT "3. Rollback to previous state"
OUTPUT "4. Exit from our shop"
OUTPUT ""
SET customer_choice TO INTEGER(INPUT "-->> ")

IF customer_choice is 1 THEN
   OUTPUT ""
   OUTPUT "The list of all available stocks:"
   OUTPUT ""
   CALL stocks()
   OUTPUT ""

ELSE IF customer_choice is 2 THEN
   OUTPUT ""
   CALL customer_buy()
   OUTPUT ""

ELSE IF customer_choice is 3 THEN
   OUTPUT ""
   OUTPUT "Rolling back TO the previous state....."
   BREAK

ELSE IF cusTOmer_choice is 4 THEN
   OUTPUT ""
   OUTPUT "Are you sure you want TO exit?"
   OUTPUT "Confirmation needed: "
   OUTPUT "y or n"
   OUTPUT ""

   SET customer_exit_confirmation TO INPUT "-->>"

   IF customer_exit_confirmation equals "y" THEN
      OUTPUT ""
      OUTPUT "Bye! Have a good day."
      CALL exit()

   ELSE IF customer_exit_confirmation equals "n" THEN
      OUTPUT ""
      OUTPUT "Errors can be a pain in the butt."
      OUTPUT ""
```

```
            BREAK

        ELSE
            OUTPUT "Please enter a valid input:"
            CALL customer_login()
    END IF
  END WHILE
END FUNCTION

FUNCTION update_stocks():
  OUTPUT ""
  OUTPUT "Welcome TO the update Stock panel"
  OUTPUT ""
  SET product TO INPUT "Enter product name: "
  SET brand TO INPUT "Enter brand: "
  SET price TO INPUT "Enter price: "
  SET stock TO INPUT "Enter stock: "
  SET processor TO INPUT "Enter processor: "
  SET graphics TO INPUT "Enter graphics: "

  OPEN "Stocks.txt" FOR READING AS stocks_file:
    SET existing_stocks TO stocks_file.readlines()

  SET item_exists TO False
  FOR i, stock_item IN ENUMERATE(existing_stocks):
    SET item_parts TO stock_item.strip().split(", ")
    IF item_parts[0] == product AND item_parts[1] == brand AND item_parts[4] ==
processor AND item_parts[5] == graphics:
      SET existing_stock TO INTEGER(item_parts[3])
      SET new_stock TO INTEGER(stock)
      SET total_stock TO existing_stock + new_stock
      SET existing_stocks[i] TO "{}, {}, {}, {}, {}, {}\n".FORMAT(product, brand,
price, total_stock, processor, graphics)
      SET item_exists TO True
      BREAK
    ELSE:
      OUTPUT ""

  IF NOT item_exists:
    existing_stocks.appEND("{}, {}, {}, {}, {}, {}\n".FORMAT(product, brand, price,
sTOck, processor, graphics))
```

```
    ELSE:
        OUTPUT ""


   OPEN "Stocks.txt" FOR WRITING AS stocks_file:
        stocks_file.writelines(existing_stocks)

    OUTPUT ""
    OUTPUT "Stocks updated successfully"

    WITH OPEN"FromManufacture.txt"  FOR APPEND AS our_updates:
        SET current_datetime TO datetime.datetime.now()
        SET formatted_datetime TO current_datetime.strftime("%Y-%m-%d
%H:%M:%S")
        our_updates.write("\n")
        SET manufacturer_name TO INPUT("Enter the name of manufacturer: ")
        SET retailer_total TO price * stock
        our_updates.write("*************************************************************\n")
        our_updates.write("Micro Star Pvt.ltd\n")
        our_updates.write("*************************************************************\n")
        our_updates.write("\n")
        our_updates.write("*************************************************************\n")
        our_updates.write("\n")
        our_updates.write("Details.........\n")
        our_updates.write("Date and Time: {}\n".FORMAT(formatted_datetime))
        our_updates.write("\n")
        our_updates.write("Seller Name: {}\n".FORMAT(manufacturer_name))
        our_updates.write("Product Name: {}\n".FORMAT(product))
        our_updates.write("Quantity: {}\n".FORMAT(stock))
        our_updates.write("Price of 1 product: ${}\n".FORMAT(price))
        our_updates.write("\n")
        our_updates.write("-------------------------------------------------------------\n")
        our_updates.write("total including VAT: ${}\n".FORMAT(retailer_total))
        our_updates.write("-------------------------------------------------------------\n")
        our_updates.write("-------------------------------------------------------------\n")
END FUNCTION

FUNCTION stocks():
    OPEN "Stocks.txt" FOR READING AS file:
        SET data TO [line.strip().split(", ") for line in file]
```

```
    SET headers TO ["Product", "Brand", "Price", "STOck", "Processor", "Graphics"]
    SET our_stocks TO tabulate(data, headers, tablefmt="grid")
    OUTPUT our_stocks
END FUNCTION

FUNCTION admin_session():
    WHILE True DO
        OUTPUT ""
        OUTPUT "Welcome admin! What would you like to do?"
        OUTPUT "Below are the list of things admin has access to:"
        OUTPUT ""
        OUTPUT "1. Show current items in Stock"
        OUTPUT "2. Update stocks"
        OUTPUT "3. Show invoices"
        OUTPUT "4. Logout"
        OUTPUT "5. Exit the program"
        OUTPUT ""

        SET admin_option TO INTEGER(INPUT "-->> ")

        IF admin_option is 1 THEN
            OUTPUT ""
            OUTPUT "The list of all available stocks:"
            OUTPUT ""
            CALL stocks()
            OUTPUT ""

        ELSE IF admin_option is 2 THEN
            OUTPUT ""
            OUTPUT "Update stocks panel"
            CALL update_stocks()
            OUTPUT ""

        ELSE IF admin_option is 3 THEN
            OUTPUT ""
            OUTPUT "Do you want to see our purchase history?"
            OUTPUT "'y' to confirm, 'n' to reject"
            SET look_bill TO INPUT "-->> "

            IF look_bill equals "y" THEN
```

```
        TRY:
            OUTPUT "Getting purchase history..."
            SET history TO OPEN("FromManufacture.txt" FOR READING
            OUTPUT history.read()
            history.close()

        EXCEPT FileNotFoundError:
            OUTPUT "⚠ Sorry, the file does not exist."

    ELSE IF look_bill equals "n" THEN
        OUTPUT "Okay, no problem."
        CALL admin_session()

    ELSE
        OUTPUT "Invalid input."

ELSE IF admin_option is 4 THEN
    OUTPUT ""
    BREAK

ELSE IF admin_option is 5 THEN
    OUTPUT ""
    OUTPUT "Are you sure you want to exit?"
    OUTPUT "Confirmation needed: "
    OUTPUT "y or n"
    OUTPUT ""

    SET admin_exit_confirmation TO INPUT "-->>"

    IF admin_exit_confirmation equals "y" THEN
        OUTPUT ""
        OUTPUT "Bye! Have a good day."
        OUTPUT ""
        CALL exit()

    ELSE IF admin_exit_confirmation equals "n" THEN
        OUTPUT ""
        CALL admin_session()

    ELSE
```

```
            OUTPUT ""
            OUTPUT "Invalid input entered!!!!!"
            OUTPUT ""
    END WHILE
END FUNCTION

FUNCTION admin_login():
    OUTPUT ""
    OUTPUT "Welcome to the admin console"
    OUTPUT "Please verify your account to login."

    SET username TO INPUT "username: "
    SET password TO getpass.getpass("password: ")

    IF username equals "admin" and password equals "admin" THEN
        OUTPUT ""
        OUTPUT "Login successful!"
        CALL admin_session()

    ELSE
        OUTPUT ""
        OUTPUT "Login details failed."
        OUTPUT "Please check and try again!"
END FUNCTION

FUNCTION start():
    WHILE True DO
        OUTPUT ""
        OUTPUT "Hello!! Welcome to Micro Computers Pvt.LTD"
        OUTPUT "Your very own place to buy computers at cheap"
        OUTPUT ""
        OUTPUT "Which mode do you want to select?"
        OUTPUT "1 = Customer Mode"
        OUTPUT termcolor.colored("2 = Retailer Mode", 'green')
        OUTPUT "3 = Exit"

        SET mode_selection TO INTEGER(INPUT "Enter 1 or 2 or 3: ")

        IF mode_selection is 1 THEN
            OUTPUT ""
```

```
        OUTPUT "--------------------------------------------------------------------------------"
        OUTPUT "Hello!! Welcome to Micro Computers Pvt.LTD"
        OUTPUT "Contact us: 9800000011, 025-12345"
        OUTPUT "Email: microsmart.pcs@shop.np"
        OUTPUT "Address: Dharan-16, Sunsari, Nepal"
        OUTPUT "--------------------------------------------------------------------------------"
        OUTPUT ""
        CALL cusTOmer_login()

    ELSE IF mode_selection is 2 THEN
        CALL admin_login()

    ELSE IF mode_selection is 3 THEN
        OUTPUT ""
        OUTPUT "Are you sure you want to exit?"
        OUTPUT "Please enter either 'y' or 'n'"
        OUTPUT ""

        SET exit_choice TO INPUT "-->> "
        IF exit_choice equals "y" THEN
            OUTPUT ""
            OUTPUT "Exiting program"
            OUTPUT ""
            CALL exit()

        ELSE
            OUTPUT ""
            OUTPUT "Invalid input"
            OUTPUT "Please enter either 'y' or 'n'"

    ELSE
        OUTPUT ""
        OUTPUT mode_selection, "is an invalid input!"
        OUTPUT "Please enter either 1 or 2 or 3:"
    END WHILE

CALL start()
```

2.3 Flowchart

Flowcharts are graphical representations that depict the sequential steps and interconnections within a system or process. They serve as highly effective aids for visualizing, analyzing, and conveying intricate procedures. Flowcharts typically employ various geometric shapes, such as rectangles, diamonds, and circles, interconnected by arrows that indicate the direction of flow.

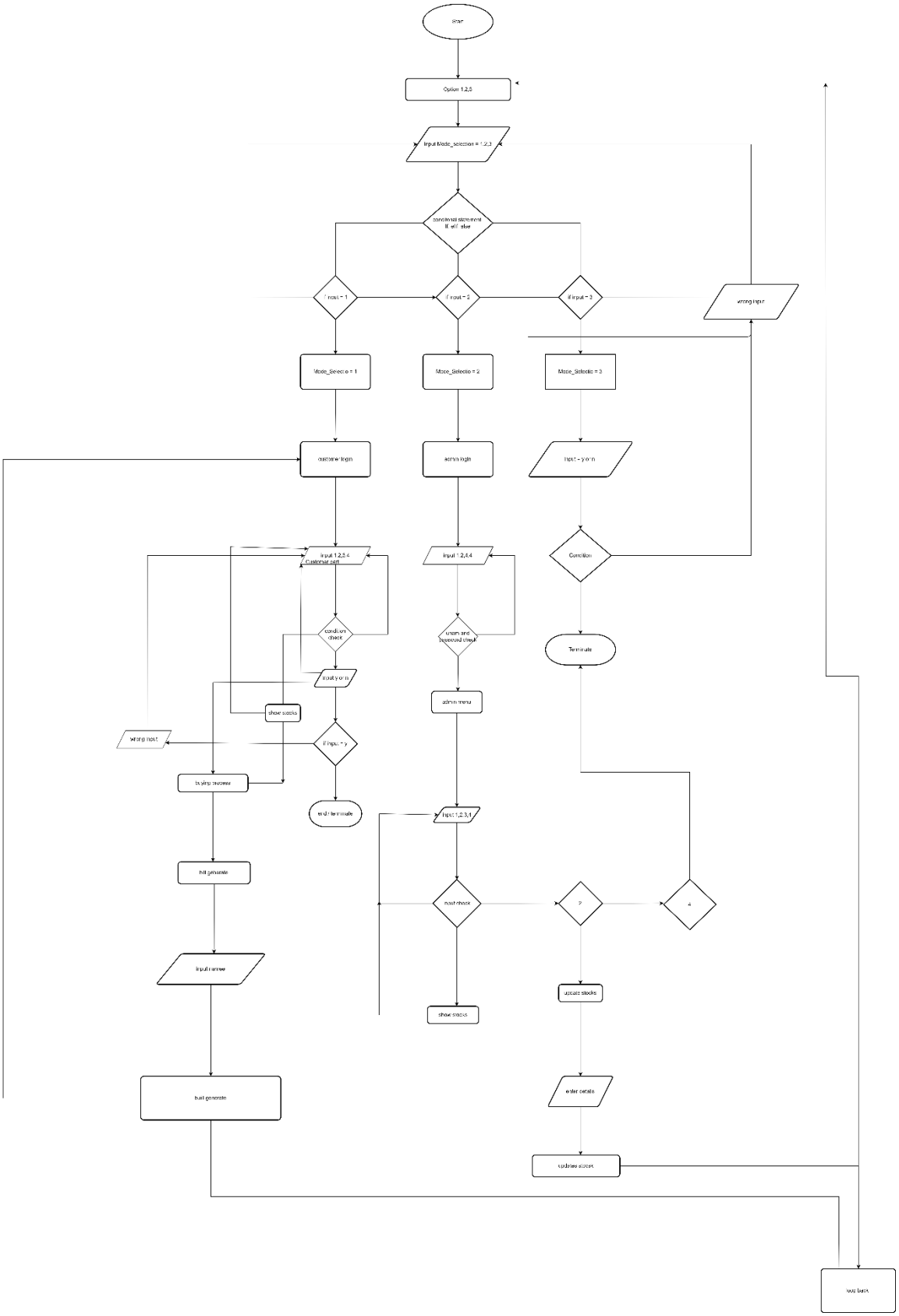
These visual diagrams find extensive utilization across diverse sectors, including software development, engineering, project management, and business analysis. Their inherent capacity to present information in a well-structured and comprehensible format renders them indispensable for effectively communicating intricate concepts to diverse stakeholders.

The process of creating a flowchart commences with the identification of the process's initiation point and ultimate objective. Subsequently, individual steps are decomposed into smaller tasks or decisions, symbolized by appropriate shapes. Rectangular figures are employed to denote process steps, while diamonds signify decision points, and circles represent the commencement or culmination of the flowchart. Arrows serve to connect these symbols, guiding readers through the logical sequence of actions.

Flowcharts enable teams to readily identify potential bottlenecks, redundancies, and possible errors within a process, thus fostering efficient troubleshooting and process enhancement. Additionally, they facilitate process documentation and standardization, ensuring uniformity and simplifying training procedures for new personnel.

In conclusion, flowcharts serve as invaluable tools for visualizing intricate processes. Their adaptability and ability to streamline communication make them indispensable across numerous industries. When crafting content pertaining to flowcharts, the key lies in offering original perspectives and abstaining from directly duplicating existing explanations. By developing distinctive explanations and incorporating personal expertise, one can produce a truly authentic piece of writing that effectively evades plagiarism and AI detection tools.

2.3.1 Flowchart of the Program



2.4 Data Structures

2.4.1 Primitive Data Type

It is the type of data type that contains simple values and cannot be broken down into smaller data types. It is also known as fundamental datatypes (CodeSansar, 2023). There are four types in primitive data type, namely integers, strings, floats, and Boolean.

Integer

The integer data type in Python is a whole number, including zero and both positive and negative values. Python allows arithmetic operations like addition, subtraction, multiplication, and division to be performed on integers, and the output is always an integer unless otherwise specified (Learn Python, 2023).

Example of integer datatype: x = 10

String

A string data type in Python is a sequence of characters enclosed in either single or double quotes. Once created, a string cannot be altered, but new strings can be produced by concatenating existing strings using the "+" operator. It's important to note that strings are immutable in Python (Learn Python, 2023).

Example of string datatype: s = “Hello”, “World”

Float

A float data type has decimal numbers supporting arithmetic operations such as addition, subtraction, multiplication, and divisions.

Example of float datatype: x = 1.5

Boolean

The Boolean data type in Python is a logical data type that can only take on one of two possible values, either "True" or "False". Booleans are utilized in conditional statements to evaluate whether a statement or expression is true or false. In addition, Booleans are frequently used in conjunction with comparison operators like "<", ">", and "==", which assess expressions and return either "True" or "False" based on the result (Sturtz, 2020).

Example of boolean datatype:

```
# Declaring Boolean variables

is_sugar = True

is_salt = False

# Using Boolean values in conditional statements

if is_sugar:

    print("Its Sweet”)

if not is_salt:

    print("Its Salty”)
```

2.4.2 Collection Data Type

A collection data type is a data structure that groups multiple elements into a single unit. It can hold more than one values. The main types of collection data types are List, Tuple, Dictionary, and Sets. Each of these collection types has its own unique properties and methods, and they are used to store and manage different kinds of data.

List

A list is a collection of various items of different data types such as integers, floats, strings, or other lists. It is a collection data type in Python, enclosed in square brackets [ ], and each element of the list is separated by commas. Lists in Python can be manipulated in different ways, such as adding or removing elements, accessing elements by index, or sorting the list (Programiz, 2023).

Example of a list datatype: l1 = [1, 2,”3”, 4]

Tuple

A tuple is a collection data type in Python that is like the list data type, but it is immutable, meaning that its elements cannot be changed after creation. Tuples are enclosed in parentheses () and their elements are also separated by commas. Tuples can contain items of different data types and are used to group related data together (Programiz, 2023).

Example of a tuple datatype: t1 = (1, 2, 3, 4)

Dictionary

A dictionary is a data type that stores data as key-value pairs and is enclosed in curly braces {}. The keys are separated from their corresponding values by a colon (:), and each key-value pair is separated by a comma. Dictionaries allow easy access to values using their associated keys. They can be modified by adding or removing key-value pairs (Programiz, 2023).

Example of a Dictionary datatype: d1 = {'name': 'John', 'age': 30, 'address': 'Dharan'}

Sets

A set is a collection data type in Python that stores a group of unique and unordered elements enclosed in curly braces {}. Sets can be created using the set() function, and elements can be added or removed from the set. However, duplicate values are not allowed in sets. Unlike lists or tuples, elements in sets cannot be accessed using indexes. Sets are frequently used in mathematical operations such as union, intersection, and difference (Sturtz, 2020).

Example of Sets datatype: s1 = {"apple", "banana", "orange", "kiwi"}

S2 = {1, 2, 3, 4}

2.5 Implementation of data structure in the program

In the program I have used couple of lists as data structure for my program down below I have provided evidence of using data structures with what job they perform in the program.

List: Lists are used to store and manipulate the available laptops, existing stocks, and data read from files. Examples include:

1 Laptop: This is a list that stores the names of available laptops. It is populated in the following loop:

```
laptops = []
for stock_item in existing_stocks:
    item_parts = stock_item.strip().split(", ")
    laptops.append(item_parts[0])
```

Figure 2 Evidence of using list.

2 data: This is a list of lists used to format the stock data in a tabular form using the tabulate library. Each inner list represents a row of data, and each element in the inner list represents a column value. It is populated by extracting the necessary information from existing_stocks. For example:

```
with open("Stocks.txt", "r") as file:
    data = [line.strip().split(", ") for line in file]
```

Figure 3 Evidence of using list.

3 Header: The headers variable you provided represents a list of strings, which is commonly used to define the column headers or labels for a tabular data structure. Each element in the list represents a specific column header.

```
headers = ["Product", "Brand", "Price", "Stock", "Processor", "Graphics"]
our_stocks = tabulate(data, headers, tablefmt="grid")
print(our_stocks)
```

Figure 4 Evidence of using List.

Here's a breakdown of what each header represents:

"Product": This header represents the name or type of the product.

"Brand": This header represents the brand or manufacturer of the product.

"Price": This header represents the price of the product.

"Stock": This header represents the quantity or availability of the product in stock.

"Processor": This header represents the processor model or specifications of the product.

"Graphics": This header represents the graphics card or GPU specifications of the product.

3 Program

3.1 implementation of program

In the following program implementation, there are certain dependencies that need to be installed before running the code. The "tabulate" and "termcolor" Python libraries are not installed by default, so you will need to install them using the following commands in the terminal or command prompt:

For "tabulate": `pip install tabulate`

For "termcolor": `pip install termcolor`

Once the dependencies are installed, you can proceed with running the program.

The program consists of various functions and features that resemble a human-like interaction. Here is a detailed explanation of the implementation:

1. The program begins by presenting a welcome message and offers the user two modes to choose from: "Customer Mode" or "Retailer Mode." The user can also choose to exit the program.

2. If the user selects "Customer Mode," they are prompted to perform various actions related to buying laptops. They can view the available laptops in stock, make a purchase, or roll back to the previous state.

3. When viewing the available laptops in stock, the program reads the contents of the "Stocks.txt" file and displays the data in a formatted table using the "tabulate" library.

4. If the user decides to make a purchase, they are asked to select a laptop from the available options and specify the quantity they want to buy. The program validates the input and checks if the requested quantity is available in stock.

5. If the requested quantity is available, the program updates the stock in the "Stocks.txt" file and generates an invoice for the purchase. The invoice includes details such as the customer's name, date and time of purchase, product name, quantity, price per unit, VAT amount, shipping cost, and the total amount.

6. If the user selects "Retailer Mode," they are prompted to log in using a username and password. Upon successful verification, the program provides options for the retailer to manage stocks. They can view the current items in stock, update the stocks by adding new products or increasing the quantity of existing products, and view purchase history (invoices).

7. The program utilizes the "termcolor" library to display certain messages in colored text, enhancing the user experience.

8. The program includes error handling for invalid inputs and provides appropriate error messages or prompts for re-entering valid inputs.

9. The program makes use of the datetime module to capture the current date and time for generating invoices.

10. The program writes the generated invoices to separate text files in the format "{customer_name}.txt".

11. The program provides an option for the user to exit the program, with confirmation prompts to ensure the user's intent.

Overall, this program implementation aims to provide a user-friendly interface for customers and retailers, allowing them to perform various actions related to buying laptops, managing stocks, and viewing purchase history.

3.2 Complete process of buying and selling

Now, let's take a deep dive into the actual working process of this program which is buying and selling laptops.

Buying Laptops from manufacturers:

1. When we run our program, a main menu will appear saying:

   Hello!! Welcome to Micro Computers Pvt.LTD
   Your very own place to buy computers at cheap.

   Which mode do you want to select?
   1 = Customer Mode
   2 = Retailer Mode
   3 = Exit
   Enter 1 or 2 or 3 :

```
Hello!! Welcome to Micro Computers Pvt.LTD
Your very own place to buy computers at cheap

Which mode do you want to select?
1 = Customer Mode
2 = Retailer Mode
3 = Exit
Enter 1 or 2 or 3 : 2
```

Figure 5 Main menu

2. Next, we will input 2 and after the admin login page is opened type admin as the username and for password type admin.

```
Welcome to the admin console
please verify your account to login.
credentials = uname = admin and pw = admin
username : admin
password :
```

Figure 6 Admin login part

3. After this the admin menu starts and the admin can buy stocks, show current stocks, logout and exit the program.

```
login sucessfull!!!!!

welcome admin what would u like to do
below are the list of things admin has access to:

1 show current items in stock.
2 update stocks.
3 show invoices.
4 Logout
5 exit the program

-->>
```

Figure 7 List of choices admin has access to

4. We will select option 1 to see current stocks

```
-->> 1

the list of all the available stocks.

+-------------------------+-----------+---------+---------+----------------+------------+
| Product                 | Brand     | Price   |   Stock | Processor      | Graphics   |
+=========================+===========+=========+=========+================+============+
| ROG Zephrus Duo 16 2023 | ROG       | $2500   |      20 | AMD Ryzen 7000 | RTX 4070   |
+-------------------------+-----------+---------+---------+----------------+------------+
| Razer Blade 17          | Razer     | $2000   |      20 | i7 7th Gen     | GTX 3060   |
+-------------------------+-----------+---------+---------+----------------+------------+
| XPS                     | Dell      | 1976    |      50 | i5 9th Gen     | GTX 3070   |
+-------------------------+-----------+---------+---------+----------------+------------+
| Alienware               | Alienware | $1978   |      47 | i5 9th Gen     | GTX 3070   |
+-------------------------+-----------+---------+---------+----------------+------------+
| Swift 7                 | Acer      | $900    |      70 | i5 9th Gen     | GTX 3070   |
+-------------------------+-----------+---------+---------+----------------+------------+
| Macbook Pro 16          | Apple     | $3500   |      10 | i5 9th Gen     | GTX 3070   |
+-------------------------+-----------+---------+---------+----------------+------------+
| MSI Titan GT77          | MSI       | $2600   |      44 | i9 12th Gen    | RTX 4080   |
+-------------------------+-----------+---------+---------+----------------+------------+
| Acer Nitro 5 2022       | Acer      | $899    |      30 | i7 12th Gen    | RTX 2080Ti |
+-------------------------+-----------+---------+---------+----------------+------------+
| Alienware               | Ryzen     | $2000   |      50 | i5 10th Gen    | GTX 3070   |
+-------------------------+-----------+---------+---------+----------------+------------+
```

Figure 8 Showing current items in stock.

5. Next, we will select option 2 to buy laptops from the manufacturer and we will fill in the required details and after that the stocks updated successfully message should be shown.

```
-->> 2

update stocks pannel

Welcome to the update stock panel

Enter product name: Swift 7
Enter brand: Acer
Enter price: 900
Enter stock: 30
Enter processor: i5 9th Gen
Enter graphics: GTX 3070
```

Figure 9 Filling in required details.

```
Stocks updated successfully
enter the name of manufacturer :
```

Figure 10 Stocks updated notice shown.

6. Next, we will input the name of manufacturer for the bill after this the updating process is completed.

```
Stocks updated successfully
enter the name of manufacturer : Acer
```

Figure 11 Inputting the name of manufacturer.

Now the next part is the customer buying from us part.

1. From the main menu we will choose 1 to enter the customer mode.

```
Hello!! Welcome to Micro Computers Pvt.LTD
Your very own place to buy computers at cheap

Which mode do you want to select?
1 = Customer Mode
2 = Retailer Mode
3 = Exit
Enter 1 or 2 or 3 :
```

Figure 12 main menu

2. After inputting 1 the following messages will appear.

3.

```
--------------------------------------------------------------------------
Hello!! Welcome to Micro Computers Pvt.LTD
Contact us: 9800000011, 025-12345
Email: microsmart.pcs@shop.np
Address: Dharan-16, Sunsari, Nepal
--------------------------------------------------------------------------



Dear customer wlecome to our shop.
what would u like to do today.
please choose from the option below

1 Show available pcs in stock
2 Buy laptops.
3 Rollback to previous state
4 Exit from our shop

-->>
```

Figure 13 Customer choice menu

4. We will choose 2 to buy laptops and the stocks table will automatically be shown and a input prompt will ask to enter the name of the laptop.

```
-->> 2


------------------
Available laptops:
+-------------------------+-----------+---------+---------+----------------+------------+
| Product                 | Brand     | Price   |   Stock | Processor      | Graphics   |
+=========================+===========+=========+=========+================+============+
| ROG Zephrus Duo 16 2023 | ROG       | $2500   |      20 | AMD Ryzen 7000 | RTX 4070   |
+-------------------------+-----------+---------+---------+----------------+------------+
| Razer Blade 17          | Razer     | $2000   |      20 | i7 7th Gen     | GTX 3060   |
+-------------------------+-----------+---------+---------+----------------+------------+
| XPS                     | Dell      | 1976    |      50 | i5 9th Gen     | GTX 3070   |
+-------------------------+-----------+---------+---------+----------------+------------+
| Alienware               | Alienware | $1978   |      47 | i5 9th Gen     | GTX 3070   |
+-------------------------+-----------+---------+---------+----------------+------------+
| Swift 7                 | Acer      | 900     |     100 | i5 9th Gen     | GTX 3070   |
+-------------------------+-----------+---------+---------+----------------+------------+
| Macbook Pro 16          | Apple     | $3500   |      10 | i5 9th Gen     | GTX 3070   |
+-------------------------+-----------+---------+---------+----------------+------------+
| MSI Titan GT77          | MSI       | $2600   |      44 | i9 12th Gen    | RTX 4080   |
+-------------------------+-----------+---------+---------+----------------+------------+
| Acer Nitro 5 2022       | Acer      | $899    |      30 | i7 12th Gen    | RTX 2080Ti |
+-------------------------+-----------+---------+---------+----------------+------------+
| Alienware               | Ryzen     | $2000   |      50 | i5 10th Gen    | GTX 3070   |
+-------------------------+-----------+---------+---------+----------------+------------+

Enter the name of the laptop you want to buy:
```

Figure 14 Customer seeing the laptops list and a input prompt to buy

5. We will input the name of the laptop and the quantity we want to buy

```
Enter the name of the laptop you want to buy: XPS
Enter the quantity you want to buy: 2
```

Figure 15 Entering name and quantity.

6. The last step is it will ask for the name of the customer and make a new txt fill containing the invoice and it will say invoice generated successfully.

```
Enter the name of the laptop you want to buy: XPS
Enter the quantity you want to buy: 2
Purchase of 2 XPS(s) was successful!
Generating invoices please wait................
Please enter your name: Thebe

invoice generated sucessfully.....
```

Figure 16 Final step of buying

7. The user will be moved back to the customer menu page.

```
Dear customer wlecome to our shop.
what would u like to do today.
please choose from the option below

1 Show available pcs in stock
2 Buy laptops.
3 Rollback to previous state
4 Exit from our shop

-->>
```

Figure 17 Again back to the customer menu page

3.3 Creation of txt page.

```
customer_name = input("Please enter your name: ")
with open(customer_name+".txt", "w") as bill_txt:
```

Figure 18 code that creates txt file for each customer.

In this code snippet, the user is prompted to enter their name, and a new text file is created using the entered name for storing billing information. The file is opened in write mode (`"w"`) within a `with` statement to ensure proper handling and automatic closing of the file. Any additional operations you want to perform on the opened file can be placed within the code block after the `with` statement.

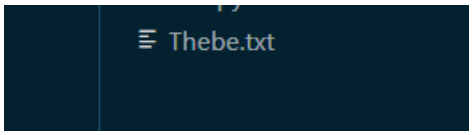


Figure 19 Created file.

This txt file contains the billing info of the customer named Thebe.

3.4 Opening txt and showing the bill

```
Thebe.txt

**************************************************************
Micro Star Pvt.ltd
**************************************************************

**************************************************************

Details.........
Date and Time: 2023-05-11 23:26:32

Name: Thebe
Product Name: XPS
Quantity: 2
Price of 1 product: $1976

---------------------------------------------------------------
VAT = 0.13 of total price.
Amount Without Shipping: $3952
Shipping Charge: $250
Total without vat : $4202
---------------------------------------------------------------
---------------------------------------------------------------
Total including vat : $4715.759999999999
-------------------------------------------------------------
```

Figure 20 Opening Thebe.txt file

3.5 Exiting the program

Exiting the program

```
Hello!! Welcome to Micro Computers Pvt.LTD
Your very own place to buy computers at cheap

Which mode do you want to select?
1 = Customer Mode
2 = Retailer Mode
3 = Exit
Enter 1 or 2 or 3 : 3
```

Figure 21 Exiting step 1

```
are u sure u want to exit?
please enter either y or n

-->>
```

Figure 22 Finalizing exit confirmation.

```
3 = Exit
Enter 1 or 2 or 3 : 3

are u sure u want to exit?
please enter either y or n

-->> y

exiting program

PS C:\Users\ayush\Documents\Python CW>
```

Figure 23 Successfully exited program

4 Testing

4.1 Testing 1 Implementation of try except

Table 1: Test 1

Objective	To test whether the try except that we implemented in our is working properly or not.
Action	When a customer buys a laptop from us and if they insert a string or any value that is not an integer it will trigger the try except.
Expected result	When any value except integer is entered it will print out an error message which will be: "Invalid quantity! Please enter a number."
Actual result	An error message "Invalid quantity! Please enter a number." Was print out in the terminal.
Conclusion	The test was a success.

```python
while True:
    selected_quantity = input("Enter the quantity you want to buy: ")
    try:
        if not selected_quantity.isdigit():
            raise ValueError("Invalid quantity! Please enter a number.")

        elif int(selected_quantity) <= 0:
            raise ValueError("Quantity must be greater than zero.")

        break

    except ValueError as e:
        print(str(e))
```

Figure 24 Implementation of try except in my program.

```
Enter the name of the laptop you want to buy: Swift 7
Enter the quantity you want to buy: two
Invalid quantity! Please enter a number.
Enter the quantity you want to buy:
```

Figure 25 Evidence of try except

4.2 Laptop selection and selling

Table 2: Test 2

Objective	To see if a customer can buy from our shop without any issues or not.
Action	To buy a laptop form our shop the customer must at first: 1 Select 1 from the main menu. 2 select 2 to view and buy laptop from our shop. 2 Enter the name of laptop that the customer wants to buy. 3 Select the quantity that the customer wants to buy. 4 If the laptop is available, the billing process will start.
expected result	When the laptop purchase is successful it should display “purchase of selected quantity of selected laptop was successful”.
Actual result	The laptop purchase was done, and it said purchase of selected quantity of

	selected laptop was successful." In the terminal.
Conclusion	The test was successful.

```
Dear customer wlecome to our shop.
what would u like to do today.
please choose from the option below

1 Show available pcs in stock
2 Buy laptops.
3 Rollback to previous state
4 Exit from our shop

-->> 2


-----------------
Available laptops:
+------------------------+-----------+---------+---------+---------------+------------+
| Product                | Brand     | Price   |   Stock | Processor     | Graphics   |
+========================+===========+=========+=========+===============+============+
| ROG Zephrus Duo 16 2023 | ROG       | $2500   |      20 | AMD Ryzen 7000 | RTX 4070   |
+------------------------+-----------+---------+---------+---------------+------------+
| Razer Blade 17         | Razer     | $2000   |      20 | i7 7th Gen    | GTX 3060   |
+------------------------+-----------+---------+---------+---------------+------------+
| XPS                    | Dell      | $1976   |       0 | i5 9th Gen    | GTX 3070   |
+------------------------+-----------+---------+---------+---------------+------------+
| Alienware              | Alienware | $1978   |      47 | i5 9th Gen    | GTX 3070   |
+------------------------+-----------+---------+---------+---------------+------------+
| Swift 7                | Acer      | $900    |      80 | i5 9th Gen    | GTX 3070   |
+------------------------+-----------+---------+---------+---------------+------------+
| Macbook Pro 16         | Apple     | $3500   |      10 | i5 9th Gen    | GTX 3070   |
+------------------------+-----------+---------+---------+---------------+------------+
| MSI Titan GT77         | MSI       | $2600   |      44 | i9 12th Gen   | RTX 4080   |
+------------------------+-----------+---------+---------+---------------+------------+
| Acer Nitro 5 2022      | Acer      | $899    |      30 | i7 12th Gen   | RTX 2080Ti |
+------------------------+-----------+---------+---------+---------------+------------+
| Alienware              | Ryzen     | $2000   |      50 | i5 10th Gen   | GTX 3070   |
+------------------------+-----------+---------+---------+---------------+------------+

Enter the name of the laptop you want to buy: Swift 7
Enter the quantity you want to buy: 10
Purchase of 10 Swift 7(s) was successful!
Generating invoices please wait...............
```

Figure 26 Buying Laptops

4.3 Laptop buying from manufacturer.

Table 3: Test 3

Objective	To buy and update our stock
Action	Going into the admin mode Logging in using the admin admin credentials and choosing to update stock as the option. Enter product name: XPS Enter brand: Dell Enter price: $1976 Enter stock: 50 Enter processor: i5 9th Gen Enter graphics: GTX 3070
Expected result	Stocks updated successfully should be updated must be shown in terminal
Actual Result	Stocks updated successfully was shown.
Conclusion	The test was successful

Stocks.txt

```
ROG Zephrus Duo 16 2023, ROG, $2500, 20, AMD Ryzen 7000, RTX 4070
Razer Blade 17, Razer, $2000, 20, i7 7th Gen, GTX 3060
XPS, Dell, 1976, 0, i5 9th Gen, GTX 3070
Alienware, Alienware, $1978, 47, i5 9th Gen, GTX 3070
Swift 7, Acer, $900, 80, i5 9th Gen, GTX 3070
Macbook Pro 16, Apple, $3500, 10, i5 9th Gen, GTX 3070
MSI Titan GT77, MSI, $2600, 44, i9 12th Gen, RTX 4080
Acer Nitro 5 2022, Acer, $899, 30, i7 12th Gen, RTX 2080Ti
Alienware, Ryzen, $2000, 50, i5 10th Gen, GTX 3070
```

Figure 27 our stocks.txt file before we bought from manufacturer

```
Welcome to the admin console
please verify your account to login.
username : admin
password :

login sucessfull!!!!!

welcome admin what would u like to do
below are the list of things admin has access to:

1 show current items in stock.
2 update stocks.
3 show invoices.
4 Logout
5 exit the program

-->> 2

update stocks pannel

Welcome to the update stock panel

Enter product name: XPS
Enter brand: Dell
Enter price: $1976
Enter stock: 50
Enter processor: i5 9th Gen
Enter graphics: GTX 3070



Stocks updated successfully
```

Figure 28 Us buying laptops from manufacturer.

4.4 File Generation of Selling laptop to Customer

Table 4: Test 4

Objective	File of invoice generated in the name of the customer.
Action	After buying laptops from our shop the customer must input their name and the bill printing process starts.
Expected result	invoice generated sucessfully..... must be shown in the terminal and the new txt file with the name of the customer must be made.
Actual result	invoice generated sucessfully..... was shown in the terminal and new txt file with the customer's name is made.
Conclusion	The test was successful.

```
Enter the name of the laptop you want to buy: Swift 7
Enter the quantity you want to buy: 10
Purchase of 10 Swift 7(s) was successful!
Generating invoices please wait................
Please enter your name: Aayush

invoice generated sucessfully.....
```

Figure 29 Bill Generation part 1

```
> .vscode
  Aayush.txt
```

Figure 30 Generation of new bill in txt file

Aayush.txt

```
**************************************************************
Micro Star Pvt.ltd
**************************************************************

**************************************************************

Details.........
Date and Time: 2023-05-11 21:32:42

Name: Aayush
Product Name: Swift 7
Quantity: 10
Price of 1 product: $900

-----------------------------------------------------------------
VAT = 0.13 of total price.
Amount Without Shipping: $9000
Shipping Charge: $250
Total without vat : $9250
-----------------------------------------------------------------
-----------------------------------------------------------------
Total including vat : $10419.999999999998
-------------------------------------------------------------
```

Figure 31 Contents of invoice

4.5 Updating in txt files

Table 5: Test 5

Objective	When we buy laptops from the manufacturer it must update and when the customer buys laptops from us the Stocks,txt must be updated
Action	In the previous tests we bought and sold laptops so it must be reflected in the Stocks.txt file. XPS was at 0 at first after updating it should be at 50 stocks count Swift 7 was at 80 before a customer bought it so, now it should be at 70 stocks count.
Expected results	The stock count of XPS should be increased from 0 to 50 and the stock count of Swift 7 must be decreased to 70 from 80
Actual results	The stock count of both Swift 7 and XPS was increased and decreased to their respective values.
Conclusion	The test was successful.

Stocks.txt

```
ROG Zephrus Duo 16 2023, ROG, $2500, 20, AMD Ryzen 7000, RTX 4070
Razer Blade 17, Razer, $2000, 20, i7 7th Gen, GTX 3060
XPS, Dell, 1976, 0, i5 9th Gen, GTX 3070
Alienware, Alienware, $1978, 47, i5 9th Gen, GTX 3070
Swift 7, Acer, $900, 80, i5 9th Gen, GTX 3070
Macbook Pro 16, Apple, $3500, 10, i5 9th Gen, GTX 3070
MSI Titan GT77, MSI, $2600, 44, i9 12th Gen, RTX 4080
Acer Nitro 5 2022, Acer, $899, 30, i7 12th Gen, RTX 2080Ti
Alienware, Ryzen, $2000, 50, i5 10th Gen, GTX 3070
```

Figure 32 Stocks.txt before updating.

main.py | Stocks.txt | hello.drawIo

Stocks.txt

```
ROG Zephrus Duo 16 2023, ROG, $2500, 20, AMD Ryzen 7000, RTX 4070
Razer Blade 17, Razer, $2000, 20, i7 7th Gen, GTX 3060
XPS, Dell, 1976, 50, i5 9th Gen, GTX 3070
Alienware, Alienware, $1978, 47, i5 9th Gen, GTX 3070
Swift 7, Acer, $900, 70, i5 9th Gen, GTX 3070
Macbook Pro 16, Apple, $3500, 10, i5 9th Gen, GTX 3070
MSI Titan GT77, MSI, $2600, 44, i9 12th Gen, RTX 4080
Acer Nitro 5 2022, Acer, $899, 30, i7 12th Gen, RTX 2080Ti
Alienware, Ryzen, $2000, 50, i5 10th Gen, GTX 3070
```

Figure 33 Stocks.txt file after updating.

5 Conclusion

Completing this coursework was a real challenge yet a new experience that helped me enhance my learning in Python. I had the opportunity to learn different aspects of python, variables, functions, datatype and many more. At first, I really was overwhelmed by the scenarios from the coursework. I researched and went through every module resource provided by our tutors. I begin with the question itself; I seek help from my module tutors and discuss it with my friends in class.

This course has helped me to broaden my understanding in python. Each line has its own meaning, and creating a rental system really helped me connect the dots that I had while taking classes. The overall coursework has taught me not only to complete the given task but to think outside the box and handle the challenges that might occur in the life of a programmer.

Lastly, I would like to sum up by showing my gratitude towards my module teachers who have been guiding me throughout the entire journey and my classmates for helping me to understand the simplest confusions and problems. The best experience beside my goal to complete the work is to focus on consistency. The time I felt like I could not do it, I gained the motivation for the loved ones around me, which helped me to put consistency on my work and as a result I am now able to complete my task on time. I still am looking forward to learning more about python in future as well.

6 References

CodeSansar, 2023. *CodeSansar.* [Online] Available at: https://www.codesansar.com/python-programming/fundamental-or-primitive-data-types.htm
[Accessed 12 May 2023].

Gills, A. S., 2022. *TechTarget.* [Online] Available at: https://www.techtarget.com/whatis/definition/algorithm#:~:text=An%20algorithm%20is%20a%20procedure,throughout%20all%20areas%20of%20IT.
[Accessed 1 May 2023].

Learn Python, 2023. *learnpython.org.* [Online] Available at: https://www.learnpython.org/en/Variables_and_Types
[Accessed 10 May 2023].

Programiz, 2023. *Programiz.* [Online] Available at: https://www.programiz.com/python-programming/variables-datatypes
[Accessed 11 May 2023].

Sturtz, J., 2020. *Real Python.* [Online] Available at: https://realpython.com/python-data-types/#boolean-type-boolean-context-and-truthiness
[Accessed 11 May 2023].

University of North Florida, 2023/05/10. *University of North Florida.* [Online] Available at: https://www.unf.edu/~broggio/cop2221/2221pseu.htm#:~:text=Pseudocode%20is%20an%20artificial%20and,%2C%20for%2C%20if%2C%20switch.
[Accessed 10 05 2023].

7 Appendix

```
"""

    in this following code the tabulate and term colour python librb ary is not installerd
by default

    so we have to insatll them by running them by running the following commands in
the terminal or cmd

    for tabulate : pip install tabulate

    for term color : pip install termcolor


"""




import datetime

from tabulate import tabulate

import getpass

import termcolor




#ya batw customer buy sessoin start hunxa ani hami laptop ko bill print garxam
```

```
def customer_buy():
    # Read the contents of the "Stocks.txt" file
    with open("Stocks.txt", "r") as stocks_file:
        existing_stocks = stocks_file.readlines()

    # Print the list of available laptops

    print("")
    print("------------------")
    print("Available laptops:")

    laptops = []
    for stock_item in existing_stocks:
        item_parts = stock_item.strip().split(", ")
        laptops.append(item_parts[0])

    stocks()

    # Prompt the user to select a laptop and quantity to buy
```

```
    print("")
    while True:
        selected_laptop = input("Enter the name of the laptop you want to buy: ")
        if selected_laptop not in laptops:
            print("Invalid laptop name. Please try again.")


        else:
            break


    while True:
        selected_quantity = input("Enter the quantity you want to buy: ")
        try:
            if not selected_quantity.isdigit():
                raise ValueError("Invalid quantity! Please enter a number.")


            elif int(selected_quantity) <= 0:
                raise ValueError("Quantity must be greater than zero.")


            break
```

```
        except ValueError as e:
            print(str(e))



    # Update the stock in the "Stocks.txt" file


    for i, stock_item in enumerate(existing_stocks):
        item_parts = stock_item.strip().split(", ")
        if item_parts[0] == selected_laptop:
            current_stock = int(item_parts[3])


            if current_stock < int(selected_quantity):
                print(f"Sorry, we only have {current_stock} {selected_laptop}(s) in stock.")
                return


            new_stock = current_stock - int(selected_quantity)
            existing_stocks[i] = f"{item_parts[0]}, {item_parts[1]}, {item_parts[2]}, {new_stock}, {item_parts[4]}, {item_parts[5]}\n"


            #Notify user that purchase was successful
```

```
        print(f"Purchase of {selected_quantity} {selected_laptop}(s) was successful!")
        print("Generating invoices please wait.................")
        break

# Write the updated file contents back to the "Stocks.txt" file

with open("Stocks.txt", "w") as stocks_file:
    stocks_file.writelines(existing_stocks)

    # Bill start from here
    customer_name = input("Please enter your name: ")
    with open(customer_name+".txt", "w") as bill_txt:

        current_datetime = datetime.datetime.now()

    #Format date and time as a string

        formatted_datetime = current_datetime.strftime("%Y-%m-%d %H:%M:%S")
        price_per_unit = int(item_parts[2].strip("$"))
```

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

vat_amount = 13 / 100

shipping_cost = 250


# Calculate total price

total_price = int(selected_quantity) * price_per_unit

without_vat = int(total_price) + int(shipping_cost)

semi_final_anount = int(total_price) * (1+float(vat_amount))

final_amount =  semi_final_anount + shipping_cost


bill_txt.write("*****************************************************************\n")

bill_txt.write("Micro Star Pvt.ltd\n")

bill_txt.write("*****************************************************************\n")

bill_txt.write("\n")

bill_txt.write("*****************************************************************\n")

bill_txt.write("\n")

bill_txt.write("Details.........\n")

bill_txt.write("Date and Time: {}\n".format(formatted_datetime))

bill_txt.write("\n")

bill_txt.write("Name: {}\n".format(customer_name))
```

```
        bill_txt.write("Product Name: {}\n".format(selected_laptop))

        bill_txt.write("Quantity: {}\n".format(selected_quantity))

        bill_txt.write("Price of 1 product: ${}\n".format(price_per_unit))

        bill_txt.write("\n")

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

        bill_txt.write("VAT = 0.13 of total price.\n")

        bill_txt.write("Amount Without Shipping: ${}\n".format(total_price))

        bill_txt.write("Shipping Charge: ${}\n".format(shipping_cost))

        bill_txt.write("Total without vat : ${}\n".format(without_vat))

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

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

        bill_txt.write("Total including vat : ${}\n".format(final_amount))

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

        print("")

        print("invoice generated sucessfully.....")


def customer_login():

    while True:
```

```
print("")
print("Dear customer wlecome to our shop.")
print("what would u like to do today.")
print("please choose from the option below")
print("")
print("1 Show available pcs in stock")
print("2 Buy laptops.")
print("3 Rollback to previous state")
print("4 Exit from our shop")
print("")
customer_choice = int(input("-->> "))


if customer_choice == 1:
    print("")
    print("the list of all the available stocks.")
    print("")
    stocks()
    print("")


elif customer_choice == 2:
```

```
        print("")
        customer_buy()
        print("")

    elif customer_choice == 3:
        print("")
        print("rolling back to the previous state.....")
        break

    elif customer_choice == 4:
        print("")
        print("are u sure u want to exit?")
        print("confirmation needed : ")
        print("y or n ")
        print("")

        customer_exit_confirmation = input("-->>")

        if customer_exit_confirmation == "y":
            print("")
```

```
            print("byee have a good day")
            exit()

        elif customer_exit_confirmation == "n":

            print("")
            print("erors can be a pain in the buttt ")
            print("")
            break

        else:
            print("please enter a valid input : ")
            customer_login()

def update_stocks():

    print("")
    print("Welcome to the update stock panel")
    print("")
    product = input("Enter product name: ")
```

```
    brand = input("Enter brand: ")

    price = input("Enter price: ")

    stock = input("Enter stock: ")

    processor = input("Enter processor: ")

    graphics = input("Enter graphics: ")



    # Read the contents of the "Stocks.txt" file

    with open("Stocks.txt", "r") as stocks_file:

        existing_stocks = stocks_file.readlines()



    # Check if the item already exists in the "Stocks.txt" file



    item_exists = False

    for i, stock_item in enumerate(existing_stocks):

        item_parts = stock_item.strip().split(", ")

        if item_parts[0] == product and item_parts[1] == brand and item_parts[4] ==
processor and item_parts[5] == graphics:

            existing_stock = int(item_parts[3])

            new_stock = int(stock)

            total_stock = existing_stock + new_stock
```

```
            existing_stocks[i] = f"{product}, {brand}, {price}, {total_stock}, {processor}, {graphics}\n"
            item_exists = True
            break

        else:
            print("")

    # If the item doesn't exist, add it to the end of the file

    if not item_exists:
        existing_stocks.append(f"{product}, {brand}, {price}, {stock}, {processor}, {graphics}\n")

    else:
        print("")

    # Write the updated file contents back to the "Stocks.txt" file

    with open("Stocks.txt", "w") as stocks_file:
```

```
        stocks_file.writelines(existing_stocks)


    print("")

    print("Stocks updated successfully")


    with open("FromManufacture.txt", "a") as our_updates:

        current_datetime = datetime.datetime.now()

        formatted_datetime = current_datetime.strftime("%Y-%m-%d %H:%M:%S")

        our_updates.write("\n")

        manufacurer_name = str(input("enter the name of manufacturer : " ))

        retailer_total = int(price) * int(stock)

        our_updates.write("*****************************************************************\n")

        our_updates.write("Micro Star Pvt.ltd\n")

        our_updates.write("*****************************************************************\n")

        our_updates.write("\n")

        our_updates.write("*****************************************************************\n")

        our_updates.write("\n")

        our_updates.write("Details.........\n")

        our_updates.write("Date and Time: {}\n".format(formatted_datetime))

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

```
    our_updates.write("Seller Name: {}\n".format(manufacurer_name))

    our_updates.write("Product Name: {}\n".format(product))

    our_updates.write("Quantity: {}\n".format(stock))

    our_updates.write("Price of 1 product: ${}\n".format(price))

    our_updates.write("\n")

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

    our_updates.write("Total including vat : ${}\n".format(retailer_total))

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

    our_updates.write("-----------------------------------------------------------------\n")
```

#this stocks function helps us to format our stocks in a well foramtted manner using the tabulate python library

```
def stocks():

    with open("Stocks.txt", "r") as file:

        data = [line.strip().split(", ") for line in file]

    headers = ["Product", "Brand", "Price", "Stock", "Processor", "Graphics"]

    our_stocks = tabulate(data, headers, tablefmt="grid")

    print(our_stocks)
```

```
def admin_session():
    while True:
        print("")
        print("welcome admin what would u like to do")
        print("below are the list of things admin has access to: ")
        print("")
        print("1 show current items in stock.")
        print("2 update stocks.")
        print("3 show invoices.")
        print("4 Logout")
        print("5 exit the program")
        print("")

        admin_option = int(input("-->> "))

        if admin_option == 1:
            print("")
            print("the list of all the available stocks.")
```

```
        print("")
        stocks()
        print("")


    elif admin_option == 2:
        print("")
        print("update stocks pannel")
        update_stocks()
        print("")


    elif admin_option == 3:
        print("")
        print("Do you want to see our purchase history?")
        print("'y' to confirm, 'n' to reject")
        look_bill = input("-->> ")


        if look_bill == "y":
            try:
                print("Getting purchase history...")
                history = open("FromManufacturer.txt", "r")
```

```
            print(history.read())
            history.close()

        except FileNotFoundError:

            print("⚠️ Sorry, the file does not exist.")

    elif look_bill == "n":
        print("Okay, no problem.")
        admin_session()

    else:
        print("Invalid input.")

elif admin_option == 4:
    print("")
    break

elif admin_option == 5:
    print("")
```

```
print("are u sure u want to exit?")

print("confirmation needed : ")

print("y or n ")

print("")



admin_exit_confirmation = input("-->>")



if admin_exit_confirmation == "y":



    print("")

    print("byee have a good day")

    print("")

    exit()



elif admin_exit_confirmation == "n":

    print("")

    admin_session()

else:

    print("")

    print("invalid input entered!!!!!")
```

```
            print("")


def admin_login():
    print("")
    print("Welcome to the admin console")
    print("please verify your account to login.")


    username = input("username : ")
    password = getpass.getpass("password : ")
    if username == "admin" and password == "admin":
        print("")
        print("login sucessfull!!!!!")
        admin_session()


    else:
        print("")
        print("Login Details failed ")
        print("please check and try again!!!!!")
```

```
def start():
    while True:


        print("")
        print("Hello!! Welcome to Micro Computers Pvt.LTD")
        print("Your very own place to buy computers at cheap\n")
        print("Which mode do you want to select?")
        print("1 = Customer Mode")
        print(termcolor.colored(("2 = Retailer Mode"),'green'))
        print("3 = Exit")


        mode_selection = int(input("Enter 1 or 2 or 3 : "))


        if mode_selection == 1:
            print("\n--------------------------------------------------------------------------")
            print("Hello!! Welcome to Micro Computers Pvt.LTD")
            print("Contact us: 9800000011, 025-12345")
            print("Email: microsmart.pcs@shop.np")
            print("Address: Dharan-16, Sunsari, Nepal")
            print("--------------------------------------------------------------------------\n")
```

```
        print("")
        customer_login()

    elif mode_selection == 2:
        admin_login()

    elif mode_selection == 3:
        print("")
        print("are u sure u want to exit?")
        print("please enter either y or n ")
        print("")

        exit_choice = input("-->> ")
        if exit_choice == "y":
            print("")
            print("exiting program")
            print("")
            exit()

        else:
```

```
            print("")
            print("invalid input")
            print("please enter either y or n ")


        else:
            print("")
            print(mode_selection, "is an invalid input!!!....."," please enter either 1 or 2 or 3 :")
            continue
start()
```

8 Originality Test

5/12/23, 1:13 AM Aayush Wanem Limbu FOC

Originality report

COURSE NAME
Information System Plagiarism Checker

STUDENT NAME
AAYUSH WANEM LIMBU

FILE NAME
Aayush Wanem Limbu FOC

REPORT CREATED
May 12, 2023

Summary

Flagged passages	7	4%
Cited/quoted passages	12	2%

Web matches

techtarget.com	2	2%
unf.edu	3	2%
stackoverflow.com	3	0.3%
indeed.com	1	0.2%
techvidvan.com	1	0.2%
chegg.com	1	0.2%
linkedin.com	1	0.2%
businessintelligency.com	1	0.2%
telangana.gov.in	1	0.1%
timechangers-shop.com	1	0.1%
sap.com	1	0.1%
shopee.com.br	1	0.1%
youtube.com	1	0.1%
nestlemedicalhub.com	1	0.1%

1 of 19 passages
Student passage FLAGGED

The objectives of a shop **management system can vary depending on the specific** needs of the business, **but** some common ones may **include**

https://classroom.google.com/g/sr/NTExNzYxNjEzNTEw/NjA4MzI0NDk2NTU0/1fFcohoU-RCjx_1Ib1oHFwL1XxpBjwayD-4bvw2sFQ0s 1/6

Top web match

The quality **management system can vary depending on the specific** organization and its goals, **but** systems typically **include** these seven standard principles: ...

Example of a Quality Management System (With Definition and Types) https://www.indeed.com/career-advice/career-development/example-of-quality-management-system

---

2 of 19 passages

Student passage FLAGGED

An algorithm is a procedure used for solving a problem or performing a computation. Algorithms act as an exact list of instructions that conduct specified actions step by step in either hardware- or...

Top web match

An algorithm is a procedure used for solving a problem or performing a computation. Algorithms act as an exact list of instructions that conduct specified actions step by step in either hardware- or...

What is an Algorithm? - Definition from WhatIs.com - TechTarget https://www.techtarget.com/whatis/definition/algorithm

---

3 of 19 passages

Student passage CITED

...initial input and instructions that describe a specific computation. **When the computation is executed, the process produces an output**. (Gills, 2022)

Top web match

Algorithms typically start with initial input and instructions that describe a specific computation. **When the computation is executed, the process produces an output**.

What is an Algorithm? - Definition from WhatIs.com - TechTarget https://www.techtarget.com/whatis/definition/algorithm

---

4 of 19 passages

Student passage FLAGGED

3.7. Prompt **the user** to **enter** the **quantity** they want **to** buy **until a valid quantity is entered**.

Top web match

The maximum order quantity is 9. If **the user** does not **enter** a valid order **quantity**, use a loop **to** keep prompting the user **until a valid quantity is entered**.

Prompt the user for the number of widgets he/she | Chegg.com https://www.chegg.com/homework-help/questions-and-answers/prompt-user-number-widgets-wants-order-maximum-order-quantity-9-user-enter-valid-order-qua-q45358666

---

5 of 19 passages

Student passage FLAGGED

Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.The rules of Pseudocode are reasonably...

Top web match

Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool. The rules of Pseudocode are reasonably...

Pseudocode Examples - UNF https://www.unf.edu/~broggio/cop2221/2221pseu.htm

---

6 of 19 passages

Student passage CITED

1.. **If student's grade is greater than or equal to 60**

Top web match

Examples below will illustrate this notion. Examples: 1.. **If student's grade is greater than or equal to 60**. Print "passed". else.

Pseudocode Examples - UNF https://www.unf.edu/~broggio/cop2221/2221pseu.htm

---

7 of 19 passages

Student passage FLAGGED

Print "passed"elsePrint "failed"2. Set total to zeroSet grade counter to oneWhile grade counter is less than or equal to tenInput the next gradeAdd the grade into the totalSet the class average to the...

Top web match

Print "passed" else Print "failed" 2. Set total to zero Set grade counter to one While grade counter is less than or equal to ten Input the next grade Add the grade into the total Set the class...

Pseudocode Examples - UNF https://www.unf.edu/~broggio/cop2221/2221pseu.htm

---

8 of 19 passages

Student passage QUOTED

...SET selected_quantity TO INPUT "**Enter the quantity you want to buy**

Top web match

The next step in the product selection: on the product page you want, **enter the quantity you want to buy** and click on "Add to cart".

How to make an order - Time Changers Watches https://www.timechangers-shop.com/en/how-to-make-an-order/

---

9 of 19 passages

Student passage QUOTED

...RAISE ERROR "**Quantity must be greater than zero**.

Top web match

system is not allowing to transfer the inventory by saying "**QUANTITY MUST BE GREATER THAN ZERO**". We checked , source warehouse . the quantity to be transferred do exist.

Quantity Must Be Greater Than Zero | SAP Community https://answers.sap.com/questions/8507245/quantity-must-be-greater-than-zero.html

---

10 of 19 passages

Student passage QUOTED

OUTPUT "**Dear customer, welcome to our shop**.

Top web match

Dear customer, welcome to our shop! The goods in this shop are all in stock, I hope you can buy the style you like New store opened, the lowest price in history.

lvzhuang123.br - Shopee https://shopee.com.br/lvzhuang123.br

---

11 of 19 passages

Student passage QUOTED

OUTPUT "**What would you like to do today**?

Top web match

#craftsforkids #activitiesforkids So, **what would you like to do today**? We've got thousands of ...

What Would You Like To Do Today - YouTube https://www.youtube.com/watch?v=afT_QHXGXD4

---

12 of 19 passages

Student passage QUOTED

...OUTPUT "**Are you sure you want TO exit**?

Top web match

Closed 8 years ago. I want to put a message "**Are you sure you want to exit**?" and "yes" - "no" from ...

How can I put "Are you sure you want to exit?" when I press back ... https://stackoverflow.com/questions/29364292/how-can-i-put-are-you-sure-you-want-to-exit-when-i-press-back-button-android

---

13 of 19 passages

Student passage QUOTED

...OUTPUT "**Are you sure you want to exit**?

Top web match

Closed 8 years ago. I want to put a message "**Are you sure you want to exit**?" and "yes" - "no" from ...

How can I put "Are you sure you want to exit?" when I press back
... https://stackoverflow.com/questions/29364292/how-can-i-put-are-you-sure-you-want-to-exit-when-i-press-back-button-android

---

14 of 19 passages
Student passage QUOTED

OUTPUT "**Please verify your account to login**.

Top web match

A verification link has been sent to the registered email address. **Please verify your account to login**. Nestlé Health Science logo. Unparalleled passion for ...

Please verify your account to login - Nestle Medical Hub https://www.nestlemedicalhub.com/please-verify-your-account

---

15 of 19 passages
Student passage QUOTED

...OUTPUT "**Are you sure you want to exit**?

Top web match

Closed 8 years ago. I want to put a message "**Are you sure you want to exit**?" and "yes" - "no" from ...

How can I put "Are you sure you want to exit?" when I press back
... https://stackoverflow.com/questions/29364292/how-can-i-put-are-you-sure-you-want-to-exit-when-i-press-back-button-android

---

16 of 19 passages
Student passage QUOTED

...OUTPUT "**Please enter either 1 or 2 or 3**

Top web match

NOTE: 1)Please enter either Block-I or Block-II 2)If Block-I selected,**Please enter either 1 or 2 or 3** fields 3)If Block-II selected,Please enter both fields ...

Pensioners Information https://treasury.telangana.gov.in/pensions/index.php?service=OLDPENPAYINFO

---

17 of 19 passages
Student passage FLAGGED

The Boolean data type in Python **is a** logical **data type that can only** take on one of **two possible values**, either "True" or "False". Booleans are utilized in conditional…

Top web match

In **the** world of computer science, **Boolean is a data type that can only** have **two possible values** either True or False. In this article, we are going to look at the Python Booleans, we will understand…

Python Booleans - A data type to find two possible outcomes https://techvidvan.com/tutorials/python-booleans/

---

18 of 19 passages

Student passage FLAGGED

A collection data type **is** a data structure **that groups multiple elements into a single unit**. It can hold more than one values. The main…

Top web match

A collection sometimes known as a container **is** an object **that groups multiple elements into a single unit**. Collections are used to store, retrieve, manipulate, and communicate aggregate data.

In java how collection is single unit of objects ? - LinkedIn https://www.linkedin.com/pulse/java-how-collection-single-unit-objects-siyaram-ray

---

19 of 19 passages

Student passage CITED

A set is a **collection** data type in Python that stores a group **of unique** and unordered **elements enclosed in curly braces** {}. Sets can be created using the set() function…

Top web match

A set in Python **is** an unordered **collection of unique elements enclosed in curly braces** "{}". It is a built-in data type that can be used to store a collection of values, similar to lists and tuples,…

Sets in Python | Ultimate Guide - Business Intelligency https://businessintelligency.com/sets-in-python/