LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4051NI Fundamentals of Computing

Assessment Weightage & Type

60% Individual Coursework

Year and Semester

2021-22 Spring

Student Name: Asmita K.C.

Group: N1

London Met ID: 21040604

College ID: NP01NT4A210021

Assignment Due Date: 13th MAY 2022

Assignment Submission Date: 13th MAY 2022

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

Table of Contents

LIST OF FIGURES

List of Tables

INTRODUCTION

PYTHON

Python is an easy-to-learn and powerful programming language. Efficient high level data structure and simple but an effective approach to object-oriented programming. With Python's elegant syntax and dynamic typing and its interpretable nature makes it an ideal language for scripting and rapid application development in more areas and most platforms (Van, et al., 1995).

Figure 1 Advantages of Python *(Ritesh, 2021)*

ABOUT THE PROJECT

This is the coursework of module fundamentals and computing all done individually with the help of software called Python. This project is concerned to a bike management system where an application is required to develop based on the quantity of the stock we got. We are required to built two different python files consisting of main function and rest of the function and is supposed to do file-handling. Inside main function, a while loop runs and every function which is inside another file is called and run in shell. In function file, different function from displaying message to invalid output is written. In this program various types of operators, data types and logics are used. Customer is asked to enter a value in which the customer can add, purchase or exit the system.

When a customer inputs any valid input, they are required to fill the given questions or inputs including name number, location and in case of the company shipping cost along with the shipping company is included. When any bike is purchased, the bike's quantity gets subtracted and is updated in the text file from which we are able to access each bike's details. When any bike is added, the bike's quantity is updated and gets added in the text file and along with this process a bill is generated with the name and number of the customer which acts as invoice. Implementation of try catch is written which helps in handling exception and some operators for checking valid input along with 2d list to store the values input from the users. Here, the loop runs until the customer enters number 3.

GOALS AND OBJECTIVES OF THE PROJECT

- To develop a python program for bike management system and present report about it.
- To construct an application to track sales, read information maintained in the text file.
- To display availability of the bikes along with the details of bike
- To write an algorithm based on this program.

DISCUSSION AND ANALYSIS

This section helps in understandning the readbility of the code through various steps such as algorithgm, flowchart and pseudocode. A flowchart is a non-technical representation of an algorithm, whereas pseudocode is a linear description of the method's main ideas. In pre-code planning, one or both of these tools can be used, depending on the situation.

ALGORITHM

An algorithm is a mathematical process to solve a problem using a finite number of steps. In the world of computers, an algorithm is the set of instructions that defines not just what needs to be done but how to do it (Techopedia, 2021)

Algorithm has the following characteristics

- Input: An algorithm may or may not require input
- Output: Each algorithm is expected to produce at least one result
- Definiteness: Each instruction must be clear and unambiguous.
- Finiteness: If the instructions of an algorithm are executed, the algorithm should terminate after finite number of steps.

Advantages of algorithm

- It is a step-wise representation of a solution to a given problem, which makes it easy to understand.
- An algorithm uses a definite procedure.
- It is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge.
- Every step in an algorithm has its own logical sequence so it is easy to debug.

ALGORITHM OF THE PROGRAM

STEP 1: Start the program

STEP 2: Display Welcome message

STEP 3: Display Bike

STEP 4: Ask User For Operation

STEP 5: loop equals True

STEP 6: START While Loop

STEP 7: Take input as user_input

STEP 8: Check IF user_input Equals to 1

STEP 9: INITIALIZE the_bike_id and the_q as 0, and bike as {}

STEP 10: START While Loop

STEP 11: Take user input from customer as name, number, email, location

STEP 12: Take valid bike input id to purchase and assign to the_bike_id

STEP 13: Take valid user input quantity to purchase and assign to the_q

STEP 14: Subtract the given quantity and update the stock

STEP 15: Calculate the price of the bike purchased

STEP 16: Write the details in bill

STEP 17: Print customer details, purchased bike details

STEP 18: Start While loop to check if user wants to purchase more

STEP 19: Take input as ask

STEP 20: Check IF ask equals yes

STEP 21: Display bike

STEP 22: Take valid bike input id to purchase and assign to the_bike_id

STEP 23: Take valid user input quantity to purchase and assign to the_q

STEP 24: Subtract the given quantity and update the stock

STEP 25: Calculate the price of the bike purchased

STEP 26: Append the details in bill with grand total

STEP 27: Print customer details, purchased bike details

STEP 28: Check IF ask equals no

STEP 29: END While Loop

STEP 30: Check for invalid input

STEP 31: Print Invalid

STEP 32: Ask User For Operation

STEP 33: END While Loop

STEP 34: Check IF user_input Equals to 2

STEP 35: INITIALIZE the_bike_id and the_q as 0, bike as {}, and bikeDetails as []

STEP 36: START While Loop

STEP 37: Take user input from customer as name, number, email, location, shipping company and cost

STEP 38: Take valid bike input id to add and assign to the_bike_id

STEP 39: Take valid user input quantity to add and assign to the_q

STEP 40: ADD the given quantity and update the stock

STEP 41: Calculate the price of the bike added

STEP 42: Write the details in invoice with shipping cost

STEP 43: Print customer details, added bike details

STEP 44: Start While loop to check if user wants to add more

STEP 45: Take input as ask

STEP 46: Check IF ask equals yes

STEP 47: Display bike

STEP 48: Take valid bike input id to add and assign to the_bike_id

STEP 49: Take valid user input quantity to add and assign to the_q

STEP 50: ADD the given quantity and update the stock

STEP 51: Calculate the price of the bike added

STEP 52: Append the details in invoice with grand total including shipping cost

STEP 53: Print customer details, added bike details

STEP 54: Check IF ask equals no

STEP 55: END While Loop

STEP 56: Check for invalid input

STEP 57: Print Invalid

STEP 58: Ask User For Operation

STEP 59: END While Loop

STEP 60: CHECK IF user_input equals to 3

STEP 61: TERMINATE The Program

STEP 62: LOOP equals False

STEP 63: END While Loop

STEP 64: Check IF user_input doesnot equals 1,2 and 3

STEP 65: PRINT Provide Integer Value

STEP 66: PROGRAM END

FLOWCHART

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams (LucidChart, 2022).

Advantages of flowchart are:

- Flowchart is an excellent way of communicating the logic of a program.
- Easy and efficient to analyze problem using flowchart.
- During program development cycle, the flowchart plays the role of a blueprint, which makes program development process easier.
- After successful development of a program, it needs continuous timely maintenance during the course of its operation. The flowchart makes program or system maintenance easier.
- It is easy to convert the flowchart into any programming language code

Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or ouptut.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.

Figure 2 Flowchart Symbols *(Elrawy, 2017)*

FLOWCHART OF THE PROGRAM

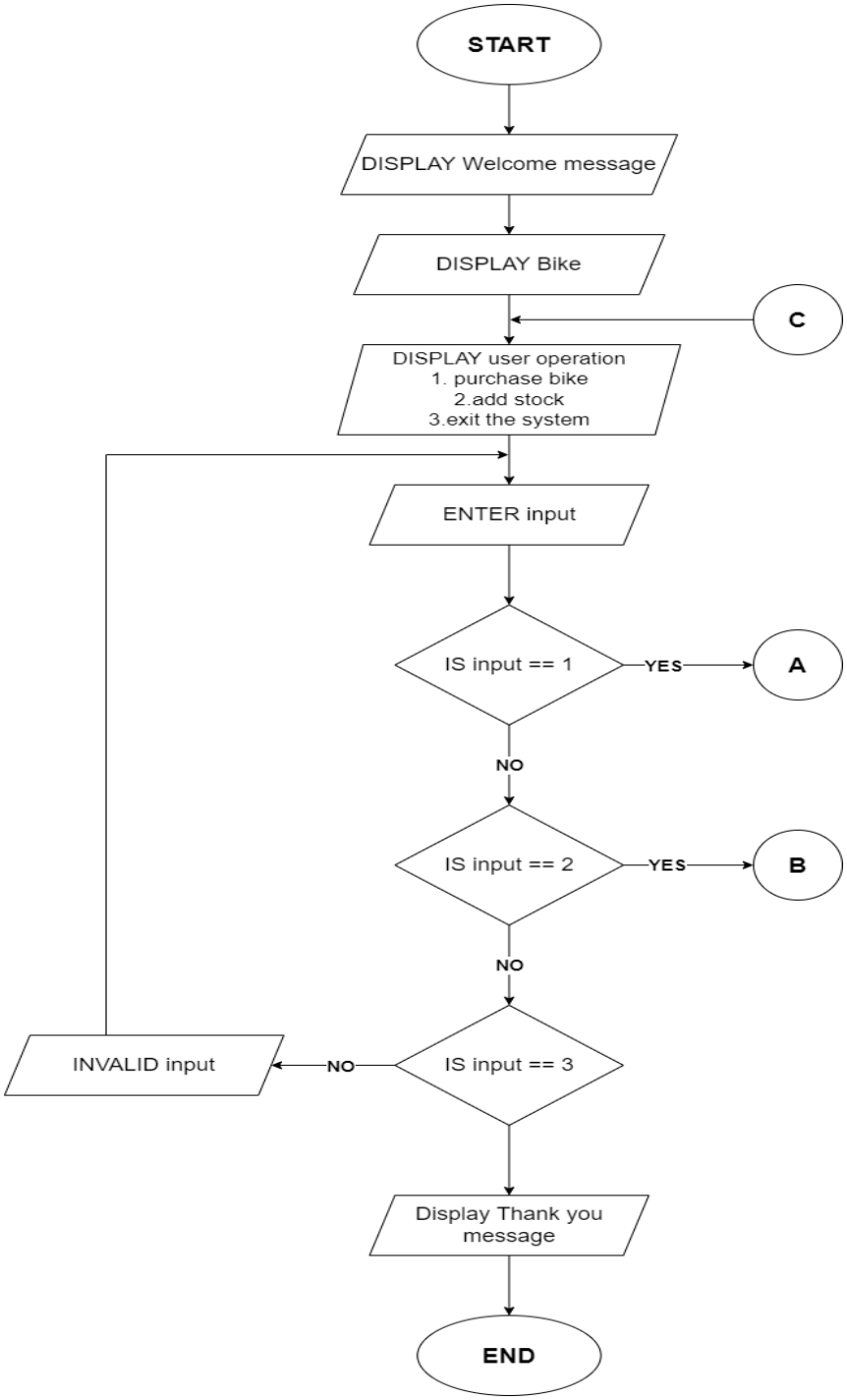


Figure 3 Flowchart of main program

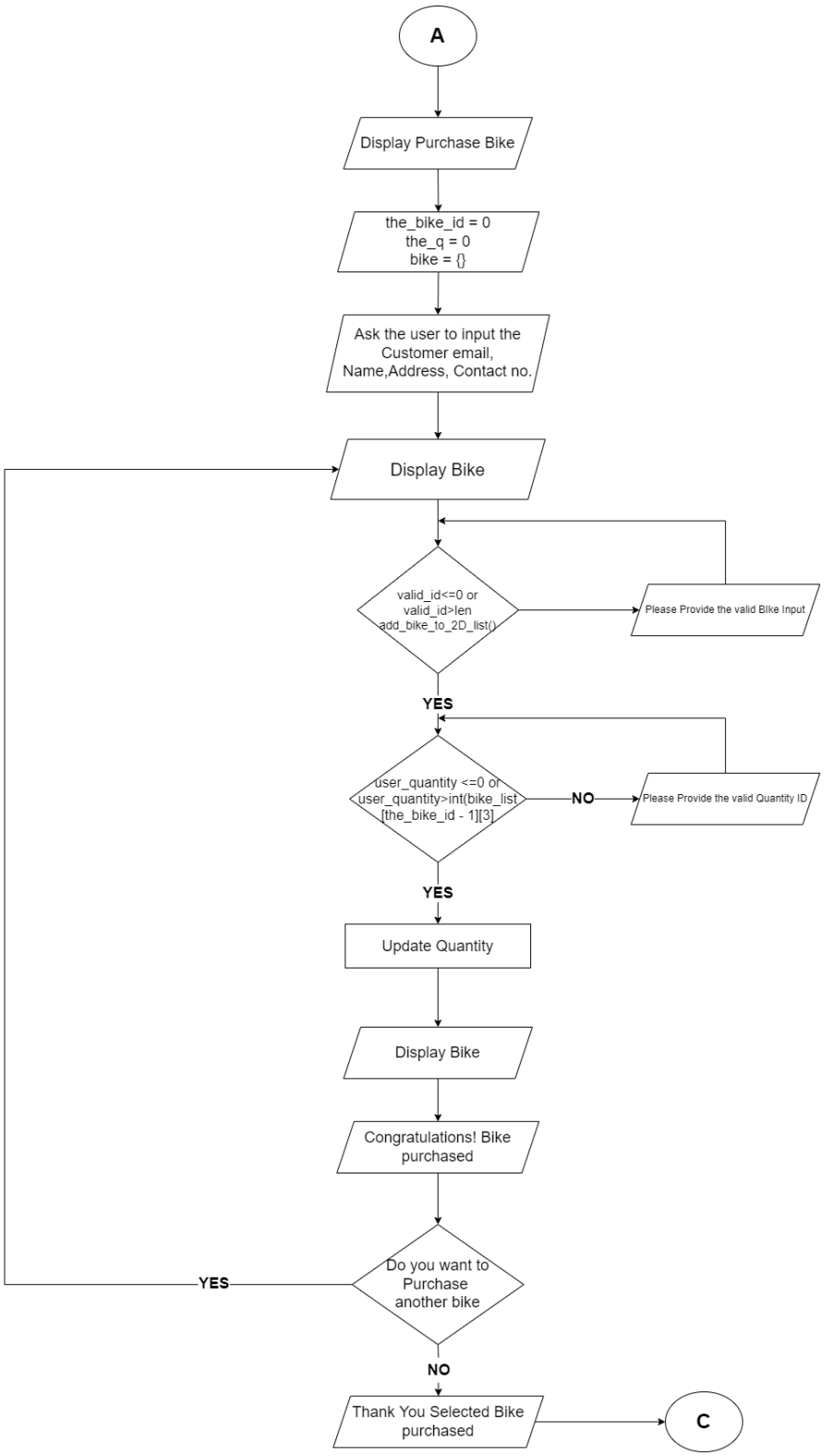


Figure 4 Flowchart if user press 1

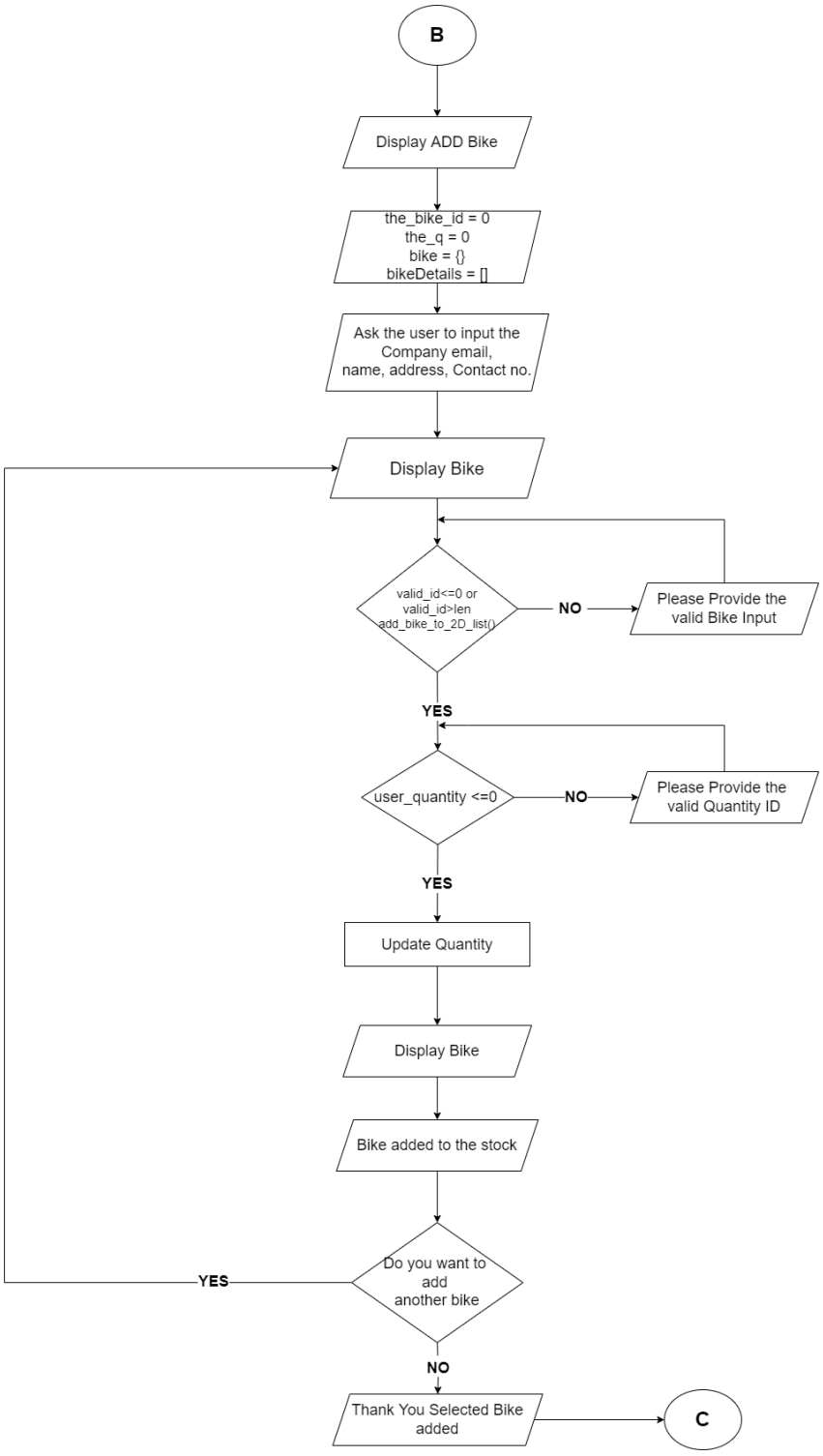


Figure 5 Flowchart if user press 2

PSEUDOCODE

Pseudo code, as the name suggests, is a false code or a representation of code which can be understood by even a layman with some school level programming knowledge (GeeksforGeeks, 2021). Pseudocode is basically an algorithm implemented using annotations and explanatory text written in simple English. Because it lacks the syntax of any programming language, it cannot be compiled or understood by a computer.

Advantages of Pseudocode are as follows:

- Improves the readability of any approach. It's one of the most effective ways to begin implementing an algorithm.
- Assists in the communication between the program and the algorithm or flowchart. Also serves as a rough manual, allowing one developer's software to be easily understood when put down in pseudo code. The documentation strategy is critical in industries. That's when a pseudo-code comes in handy.
- A pseudo code's major objective is to describe what each line of a program should accomplish, making the code building step easier for the programmer.

PSEUDO-CODE FOR WELCOME FUNCTION

```
FUNCTION welcome()
    DO
        PRINT "Welcome to the program"
    END DO
END FUNCTION
```

PSEUDO-CODE FOR DISPLAY BIKE FUNCTION

```
FUNCTION display_bike()
    DO
        PRINT Bike-details
        START TRY BLOCK
            OPEN file bike.txt in READ mode
            ASSIGN value 1 to a
            FOR line in file:
                PRINT a
                INCREASE value of a by 1
                CLOSE file
            END FOR
        END TRY BLOCK
        EXCEPT IOError
            PRINT File name misplaced!
        END EXCEPT
    END DO
END FUNCTION
```

PSEUDO-CODE FOR ADD BIKE 2D LIST FUNCTION

```
FUNCTION add_bike_2D_list()
    DO
        READ file bike.txt
        INITIALIZE my_list as [ ]
        FOR i in read_file
            APPEND my_list and split (“,”)
        RETURN my_list
        END FOR
    END DO
END FUNCTION
```

PSEUDO-CODE FOR USER OPERATION FUNCTION

```
FUNCTION user_operation()
    DO
        PRINT Enter 1 to purchase the bike
        PRINT Enter 2 to add stock
        PRINT Enter 3 to exit
    END DO
END FUNCTION
```

PSEUDO-CODE FOR ADD BIKE STOCK FUNCTION

```
FUNCTION add_bike_stock()
    DO
        PRINT Bike has been added to stock
    END DO
END FUNCTION
```

PSEUDO-CODE FOR VALIDATING BIKE ID FUNCTION

```
FUNCTION validating_bike_id()
    DO
        SET loop to True
        WHILE loop equals True
            START TRY BLOCK
                DECLARE valid_id as global
                GET valid_id as input of wanted bike id
                WHILE valid_id <= 0 OR
                 valid_id > add_bike_2D_list().count - 1
                    PRINT Provide valid Bike ID
                    CALL FUNCTION display_bike()
                    GET valid_id input
                END WHILE
                RETURN the value of valid_id
                SET loop equals to False
            END TRY BLOCK
            EXCEPT ValueError
                PRINT Enter integer value
            END EXCEPT
        END WHILE
    END DO
END FUNCTION
```

PSEUDO-CODE FOR PURCHASE BIKE FUNCTION

```
FUNCTION purchase_bike ()
    DO
        PRINT Bike has been purchased
    END DO
END FUNCTION
```

PSEUDO-CODE FOR INVALID USER INPUT FUNCTION

```
FUNCTION invalid_user_input()
    DO
        PRINT Invalid Input
    END DO
END FUNCTION
```

PSEUDO-CODE FOR EXIT FUNCTION

```
FUNCTION exit_system()
    DO
        PRINT Thank you for using our system
    END DO
END FUNCTION
```

PSEUDO-CODE FOR UPDATE STOCK FUNCTION

```
FUNCTION update_stock( bike_list )
    DO
        OPEN file bike.txt in write mode
        FOR i in bike_list
            WRITE updated stock in file
        END FOR
        CLOSE file
        CALL FUNCTION display_bike()
    END DO
END FUNCTION
```

PSEUDO-CODE FOR ADD STOCK FUNCTION

```
FUNCTION add_stock( bike_list )
    DO
        OPEN file bike.txt in write mode
        FOR i in bike_list
            WRITE added stock in file
        END FOR
        CLOSE file
        CALL FUNCTION display_bike()
    END DO
END FUNCTION
```

PSEUDO-CODE FOR QUANTITY VALIDATION FUNCTION

```
FUNCTION quantity_validation(the_bike_id)
    DO
        SET loop to True
        WHILE loop EQUALS True
            START TRY BLOCK
                SET bike_list to add_bike_2D_list()
                GET integer user_quantity input to purchase bike quantity
                WHILE user_quantity <=0 or
                user_quantity > int (bike_list [the_bike_id - 1][3]):
                    PRINT Provide Valid Quantity
                    GET integer user_quantity input to purchase bike
                    quantity
                    CALL function display_bike()
                END WHILE
                RETURN the value of user_quantity
            END TRY BLOCK
            EXCEPT ValueError
                PRINT Only Integer Values
            SET loop to False
        END WHILE
    END DO
END FUNCTION
```

PSEUDO-CODE FOR QUANTITY VALIDATION FOR BIKE ADD FUNCTION

```
FUNCTION quantity_val(the_bike_id)
    DO
        SET loop to True
        WHILE loop EQUALS True
            START TRY BLOCK
                DECLARE bike_list and id as global
                SET bike_list to add_bike_2D_list()
                SET id as the_bike_id
                GET integer user_quantity input to purchase bike quantity
                WHILE user_quantity <=0
                    PRINT Provide Valid Quantity
                    GET integer user_quantity input to purchase bike
                    quantity
                    CALL function display_bike()
                END WHILE
                RETURN the value of user_quantity
            END TRY BLOCK
            EXCEPT ValueError
                PRINT Only Integer Values
            SET loop to False
        END WHILE
    END DO
END FUNCTION
```

PSEUDO-CODE FOR FINAL SELL FUNCTION

```
FUNCTION final_sell (user_input_bike_id, the_quantity)
    DO
        SET bike_list to add_bike_2D_list()
        bike_list [user_input_bike_id - 1] [3] =
        integer (bike_list [user_input_bike_id - 1] [3]) - the_quantity
        CALL function update_stock (bike_list)
    END DO
END FUNCTION
```

PSEUDO-CODE FOR FINAL ADD FUNCTION

```
FUNCTION final_add (user_input_bike_id, the_quantity)
    DO
        SET bike_list to add_bike_2D_list()
        bike_list [user_input_bike_id - 1] [3] =
        integer (bike_list [user_input_bike_id - 1] [3]) + the_quantity
        CALL function add_stock (bike_list)
    END DO
END FUNCTION
```

PSEUDO-CODE FOR TOTAL PRICE FUNCTION

```
FUNCTION total_price(user_input_bike_id, the_quantity)
    DO
        SET bike_list to add_bike_2D_list()
        total_price =
        integer (bike_list[user_input_bike_id - 1] [4]. replace ("$","")) * the_quantity
        RETURN the value of total_price
    END DO
END FUNCTION
```

PSEUDO-CODE FOR COLOR FUNCTION

```
FUNCTION color (user_input_bike_id)
    DO
        SET bike_list to add_bike_2D_list()
        bcolor = (bike_list [user_input_bike_id - 1] [2])
        PRINT bike color as bcolor
        RETURN bcolor
    END DO
END FUNCTION
```

PSEUDO-CODE FOR BIKE NAME FUNCTION

```
FUNCTION nameb (user_input_bike_id)
    DO
        SET bike_list to add_bike_2D_list()
        bname = (bike_list [user_input_bike_id - 1] [0])
        PRINT bike name as bname
        RETURN bname
    END DO
END FUNCTION
```

PSEUDO-CODE FOR BIKE COLOR FUNCTION

```
FUNCTION cname (user_input_bike_id)
    DO
        SET bike_list to add_bike_2D_list()
        cname = (bike_list [user_input_bike_id - 1] [1])
        PRINT bike's company name as cname
        RETURN cname
    END DO
END FUNCTION
```

PSEUDO-CODE FOR BIKE PRICE FUNCTION

```
FUNCTION price (user_input_bike_id)
    DO
        SET bike_list to add_bike_2D_list()
        bprice = (bike_list [user_input_bike_id - 1] [4]. replace ("$",""))
        PRINT price of bike as bprice
        RETURN bprice
    END DO
END FUNCTION
```

PSEUDO-CODE FOR PRICE VALIDITY FUNCTION

```
FUNCTION validPrice (text)
    DO
        SET price to input(text)
        START TRY BLOCK
            price = integer (price. replace ("$"," "))
        END TRY BLOCK
        EXCEPT
            PRINT Price contains only numbers and $ sign
            price = validPrice(text).replace('$','')
        END EXCEPT
        RETURN string price with $
    END DO
END FUNCTION
```

PSEUDO-CODE FOR NUMBER VALIDITY FUNCTION

```
FUNCTION number (text)
    DO
        SET in_num to input(text)
        START TRY BLOCK
            in_num = integer(in_num)
        END TRY BLOCK
        EXCEPT
            PRINT Phone number contains only numbers
            in_num = number(text)
        END EXCEPT
        RETURN string in_num
    END DO
END FUNCTION
```

PSEUDO-CODE FOR NAME VALIDITY FUNCTION

```
FUNCTION check_name (text)
    DO
        SET in_name to input(text)
        START TRY BLOCK
            IF in_name is not alphabet
                RAISE ValueError
            IF END
        END TRY BLOCK
        EXCEPT ValueError
            PRINT Name contains only alphabets
            in_num = check_name(text)
        END EXCEPT
        RETURN in_name
    END DO
END FUNCTION
```

PSEUDO-CODE FOR USER INPUT FUNCTION

```
FUNCTION user()
    DO
        DECLARE name, num, email, location, curr_date as global
        INPUT customer name as name
        INPUT phone number as num
        INPUT email address as email
        INPUT location as location
        date = datetime.datetime.now ()
        GET curr_date in month-day-year hour-minute-second format
        PRINT Welcome to our Bike Store
    END DO
END FUNCTION
```

PSEUDO-CODE FOR COMPANY USER INPUT FUNCTION

```
FUNCTION com_user()
    DO
        DECLARE comp_name, comp_number, comp_email, comp_location,
        current_date, ship, shipCostas global
        INPUT company name as comp_name
        INPUT phone number as comp_num
        INPUT email address as comp_email
        INPUT location as comp_location
        date = datetime.datetime.now ()
        GET curr_date in month-day-year hour-minute-second format
        INPUT shipping company as ship
        INPUT shipping cost as shipCost
        PRINT Welcome to our Bike Store
    END DO
END FUNCTION
```

PSEUDO-CODE FOR USER DETAIL PRINT FUNCTION

```
FUNCTION puser()
    DO
        PRINT Full Name
        PRINT Phone Number
        PRINT Email-address
        PRINT Current Location
        CALL function nameb(the_bike_id)
        CALL function cname(the_bike_id)
        CALL function color(the_bike_id)
        PRINT Quantity of Bike
        CALL function price(the_bike_id)
        PRINT Total cost of bike
        PRINT Date
    END DO
END FUNCTION
```

PSEUDO-CODE FOR USER DETAIL TO ADD STOCK PRINT FUNCTION

```
FUNCTION comp_user()
    DO
        PRINT Company Name
        PRINT Phone Number
        PRINT Email-address
        PRINT Current Location
        CALL function nameb(the_bike_id)
        CALL function cname(the_bike_id)
        CALL function color(the_bike_id)
        PRINT Quantity of Bike
        CALL function price(the_bike_id)
        PRINT Total cost of bike
        PRINT Name of shipping company
        PRINT Shipping Cost
        PRINT Date
    END DO
END FUNCTION
```

PSEUDO-CODE FOR USER TO PURCHASE MORE FUNCTION

```
FUNCTION user_ask()
    DO
        SET run as True
        WHILE run equals True
            ask = input ("Do you want to purchase another bike? Y/N")
            IF ask equals Y
                CALL function display_bike()
                SET the_bike_id as validating_bike_id()
                SET the_q as quantity_validation(the_bike_id)
                CALL function final_sell(the_bike_id, the_q)
                SET the_price as total_price(the_bike_id, the_q)
                UPDATE bike with ({the_bike_id:the_q})
                APPEND bill with the_bike_id, bike, name, num, email,
                 location, curr_date
                CALL function purchase_bike()
                CALL function puser()
            ELIF ask equals N
                SET run as False
            ELSE
                PRINT Invalid Input
            ENDIF
```

```
            END WHILE
        END DO
END FUNCTION
```

PSEUDO-CODE FOR BIKE PURCHASE FUNCTION

```
FUNCTION purchaseBike ()
    DO
        DECLARE the_bike_id, bike, the_q, the_price as global
        PRINT Purchase Bikes
        INITIALIZE the_bike_id and the_q as 0 and bike as { }
        SET buyMore as True
        WHILE buyMore
            CALL function user()
            SET the_bike_id as validating_bike_id()
            SET the_q as quantity_validation(the_bike_id)
            CALL function final_sell(the_bike_id, the_q)
            SET the_price as total_price(the_bike_id, the_q)
            WRITE bill with the_bike_id, bike, name, num, email,
             Location, curr_date
            CALL function purchase_bike()
            CALL function puser()
            CALL function user_ask()
            CALL function user_operation()
            SET buyMore as False
        END WHILE
    END DO
END FUNCTION
```

PSEUDO-CODE FOR ADD MORE BIKE FUNCTION

```
FUNCTION comp_ask ()
    DO
        SET run as True
        WHILE run equals True
            ask = input ("Do you want to purchase another bike? Y/N")
            IF ask equals Y
                CALL function display_bike()
                SET the_bike_id as validating_bike_id()
                SET the_q as quantity_validation(the_bike_id)
                CALL function final_add(the_bike_id, the_q)
                SET the_price as total_price(the_bike_id, the_q)
                UPDATE bikeDetails = [the_q,ship,shipCost.replace('$','')]
                UPDATE bike with ({the_bike_id:bikeDetails})
                APPEND invoice with the_bike_id, bike, comp_name,
                comp_number, comp_email, comp_location, current_date,
                 ship, shipCost
                CALL function add_bike_stock()
                CALL function comp_user()
            ELIF ask equals N
                PRINT Thank You
                SET run as False
```

```
                ELSE
                    PRINT Invalid Input
                ENDIF
            END WHILE
        END DO
END FUNCTION
```

PSEUDO-CODE FOR STOCK ADD FUNCTION

```
FUNCTION addStock ()
    DO
        DECLARE the_bike_id, bike, the_q, the_price as global
        PRINT Add Stock
        INITIALIZE the_bike_id and the_q as 0 and bike as { } and
        bikeDetails as []
        SET addMore as True
        WHILE addMore
            CALL function com_user()
            SET the_bike_id as validating_bike_id()
            SET the_q as quantity_validation(the_bike_id)
            CALL function final_sell(the_bike_id, the_q)
            SET the_price as total_price(the_bike_id, the_q)
            UPDATE bikeDetails = [the_q,ship,shipCost.replace('$','')]
            CALL function add_bike_to_stock()
            UPDATE bike with ({the_bike_id:bikeDetails})
            WRITE invoice with the_bike_id, bike, comp_name,
            comp_number, comp_email, comp_location, current_date,
            ship, shipCost
            CALL function comp_user()
            CALL function comp_ask()
            CALL function user_operation()
```

```
                    SET addMore as False
            END WHILE
        END DO
END FUNCTION
```

PSEUDO-CODE FOR GENERATING PURCHASE BILL FUNCTION

```
FUNCTION bill(the_bike_id,bike, name, num, email, location, curr_date)
    DO
        DECLARE bike_id and b_curr as global
        ASSIGN results as add_bike_2D_list()
        ASSIGN bike_id as the_bike_id
        ASSIGN b_name as name
        ASSIGN b_num as num
        ASSIGN b_email as email
        ASSIGN b_location as location
        ASSIGN b_curr as curr_date
        ASSIGN bbike as bike
        GET date and time
        INITIALIZE grandTotal as 0
        OPEN file with invoice + name+ number in txt form as write
            WRITE Bike Management System
            WRITE Invoice date and time
            WRITE customer name, phone number, email-address, current
             Location
        END OPEN
        FOR i in bbike
            OPEN file with invoice + name+ number in txt form as
            append
```

```
                    WRITE bike name , color, quantity, total amount
            END OPEN
            grandTotal = grandTotal + int(results[i-1][4].replace('$',''))*bbike[i]
        END FOR
        OPEN file with invoice + name+ number in txt form as append
            WRITE grandTotal
        END OPEN
    END DO
END FUNCTION
```

PSEUDO-CODE FOR GENERATING ADD BILL FUNCTION

```
FUNCTION invoice (the_bike_id,bike, comp_name, comp_number, comp_email,
comp_location, current_date, ship, shipCost)
    DO
        DECLARE bike_id and dd as global
        ASSIGN results as add_bike_2D_list()
        ASSIGN bike_id as the_bike_id
        ASSIGN nam as comp_name
        ASSIGN numb as comp_number
        ASSIGN em as comp_email
        ASSIGN loc as comp_location
        ASSIGN dd as current_date
        ASSIGN sship as ship
        ASSIGN sshipCost as shipCost
        ASSIGN bbike as bike
        GET date and time
        INITIALIZE grandTotal as 0
        OPEN file with invoice + comp_name+ comp_number in txt form as write
            WRITE Bike Management System
            WRITE Invoice date and time
            WRITE customer name, phone number, email-address, current
             Location, shipping company, shipping cost
        END OPEN
```

```
        FOR i in bbike
            OPEN file with invoice + comp_name+ comp_number in txt form as
            append
                WRITE bike name , color, quantity, amount, shipping
                company, shipping cost
            END OPEN
            grandTotal = grandTotal + int(results[i-1][4].replace('$',''))*bbike[i][0]
                        + int(bbike[i][2])
        END FOR
        OPEN file with invoice + comp_name+ comp_number in txt form as
        append
            WRITE grandTotal
        END OPEN
    END DO
END FUNCTION
```

PSEUDO-CODE FOR MAIN FUNCTION

```
IMPORT function
CALL function welcome()
CALL function display_bike()
CALL function ask_user_for_operation()

FUNCTION check_user_input()
    DO
        loop = True
        WHILE loop equals True:
            TRY BLOCK:
                user_input = input
                IF user_input equals 1
                    CALL FUNCTION purchaseBike()
                ELIF user_input equals 2
                    CALL function addStock()
                ELIF user_input equals 2
                    CALL function exit_system()
                    loop = False
                ELSE
                    CALL function invalid_user_input()
                END IF
            END TRY BLOCK
```

```
            EXCEPT ValueError
                PRINT Provide Integer Value
            END EXCEPT
        END WHILE
    END DO
END FUNCTION
```

DATA STRUCTURES

The fundamental components around which you build your programs are data structures. Depending on your use case, each data structure provides a unique manner of arranging data so that it can be accessible quickly. Python's standard library includes a large number of data structures (Real Python, 2020).

Advantages Of Data Structures

- Data structures enable the storage of information on hard disks.
- It helps in managing large data sets.
- It plays a significant role on designing algorithm.
- It secures the data and can’t be lost. One can use the stored data in multiple projects and programs.
- It processes the data easily.

“Algorithm + Data Structure = Program” (Writh, 1976)

To develop a program of an algorithm, we should select an appropriate data structure for that algorithm. Therefore, algorithm and its associated data structures form a program.

There are two kind of data types, which are further divides into subtypes:

- Primitive data types
- Non-primitive data types

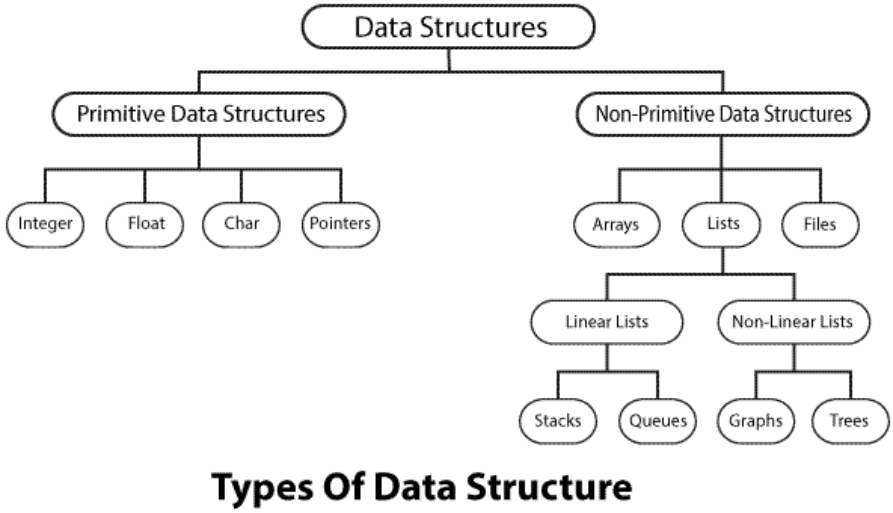


Figure 6 Types of Data Structure *(TUTORIALINK, 2022)*

Primitive Data Structures

Primitive data types are the pre-defined data types, which are supported by a programming language. These are used by programmers during the creation of new variables. Examples of primitive data types are: integer, float, double, string, boolean etc.

- **Integers**

  This value is represented by int class. It is used to represent numeric data, more specifically whole numbers from negative infinity to infinity,like 2,3, -100000, 10000 (Kilonzi, 2020). Generally, int is preferred data type when you create variables with numeric value.

For example:

```
global valid_id
valid_id = int(input("Enter the ID of the bike you want: "))
print("\n")
```

Figure 7 int data type used in program

- **String**

  Strings are collection of alphabets, words or other characters. In Python, strings are created by enclosing a sequence of characters within a pair of double or single quotes. To concatenate two or more Strings, the '+' operation can be applied to them. Repeating, splicing, capitalizing, and retrieving are some of the other String operations in Python. It is represented by str class.

  For example:

```
dt = str(datetime.datetime.now().year) + "-" + str(datetime.datetime.now().month) + "-" + str(
    datetime.datetime.now().day) + "-" + str(datetime.datetime.now().hour) + "-" + str(
    datetime.datetime.now().minute) + "-" + str(datetime.datetime.now().second)
invoice = str(dt)  # unique invoice
t = str(datetime.datetime.now().year) + "-" + str(datetime.datetime.now().month) + "-" + str(
    datetime.datetime.now().day)  # date
d = str(t)  # date
u = str(datetime.datetime.now().hour) + ":" + str(datetime.datetime.now().minute) + ":" + str(
    datetime.datetime.now().second)  # time
```

Figure 8 String data type in program while extracting date and time

```
comp_email = input("Enter company email address: ")
comp_location = input("Enter the company current location: ")
```

Figure 9 Use of string data type

- **Float**

  Float stands for floating point number. It is used for rational numbers usually ending with decimal figure such as 1.1, 2.9,5.6 etc. Since Python is a dynamically typed programming language, the data type that an object stores is mutable, and there is no need to state the type of your variable explicitly (UpGrad, 2020).

- **Boolean**

  It is a built-in data type that can take the values TRUE or FALSE. This data type is used to track **true/false conditions**. And most useful in looping while cases. . Booleans are useful in conditional and comparison expressions, for example 3 > 2 = True.

  For example,

```
run = True
while run == True:
    ask = input("Do you w
```

Figure 10 Boolean data structure in program

Non-Primitive Data Structures

The data types that are derived from primary data types are known as non-Primitive data types. These datatypes are used to store group of values. Non-primitive types are a more advanced data structure family since they hold a group of values rather than a single item. They may be divided into two types: built-in and user-defined structures. Python provides implicit support for the following built-in structures:

- **List**

  This is Python's most flexible data structure, and it's represented as a list of comma-separated components contained in square brackets. Both heterogeneous and homogeneous entries can be found in a List. Index(), append(), extend(), insert(), delete(), pop(), and other methods are available on a List. Lists are changeable, which means that their content may be modified while maintaining their identity. Lists are used to reserve/store the data of various data types in a subsequent way. Every element of the list has an address which we can call the index of an element. It starts from 0 and ends at the last element. For notation, it is like (0, n-1). It also supports negative indexing, which starts at -1 and allows us to explore the items from end to beginning.

  For example,

```
def add_bike_2D_list():
    read_file = open("bike.txt","r")
    my_list = []
    for i in read_file:
        i = i.replace("\n","")
        my_list.append(i.split(","))
    return my_list
```

List

Figure 11 List declare in program

```
bike = {}
bikeDetails = []
addMore = True
while addMore:
    com_user()
    the_bike_id = validating_bike_id()
    the_q = quantity_val(the_bike_id)
    final_add(the_bike_id, the_q)
    the_price = total_price(the_bike_id, the_q)
    bikeDetails = [the_q,ship,shipCost.replace('$','')]
    add_bike_stock()
```

List Declare

List Update

Figure 12 List declare and update in program

- **Tuples**

Tuples are immutable and comparable to Lists. Tuples are also declared in parenthesis rather than square brackets, unlike Lists. Immutability means that once an element in a Tuple has been specified, it cannot be removed, reallocated, or modified. It guarantees that the data structure's specified values are neither changed or overwritten.

For example,

```
tuple = ("Key", 16, True, 20,"Mist") #tuple with string,integer and boolean values
print(tuple)

========== RESTART: C:\Users\
('Key', 16, True, 20, 'Mist')
```

Figure 13 Tuples example

- **Dictionaries**

Dictionaries consist of key / value pairs. The "key" identifies the element and the "value" stores the value of the element. The colon separates the key from its value. The elements are separated by commas and are enclosed in curly braces throughout. The key is immutable (number, string, or tuple), but the value can be of any type. It is similar to an address book in which we may find a person's address or contact information by knowing only his or her name, i.e. we associate keys (names) with values (information).

For example,

```
def purchaseBike():
    global the_bike_id, bike, the_q, the_price
    print("\n\n----------------------")
    print("|  Purchase  Bikes   |")
    print("----------------------\n")
    the_bike_id = 0
    the_q = 0
    bike = {}                                  Dictionary
    buyMore = True
    while buyMore:
        user()
        the_bike_id = validating_bike_id()
        print("\n")
        the_q = quantity_validation(the_bike_id)
        bike.update({the_bike_id:the_q})       Dictionary update

        final_sell(the_bike_id, the_q)
        the_price = total_price(the_bike_id, the_q)
        purchase_bike()
        bill(the_bike_id, bike, name, num, email, location, curr_date)
```

Figure 14 Declaring dictionary and updating in program

- **Set**

Sets are an unordered collection of unique elements. Sets are mutable but can hold only unique values in the dataset. Set operations are similar to the ones used in arithmetic. Some Set methods include count(), index(), any(), all(), etc. Sets don't support any slicing or indexing operations because their items aren't indexed.

For example,

```
sets = {1, 2, 3, 4, 3, 2} #set with duplicates but sets cannot hold
print(sets)

sets.add("Python") #adding element into set
print(sets)

sets.remove("Python")#removing element from set
print(sets)
```

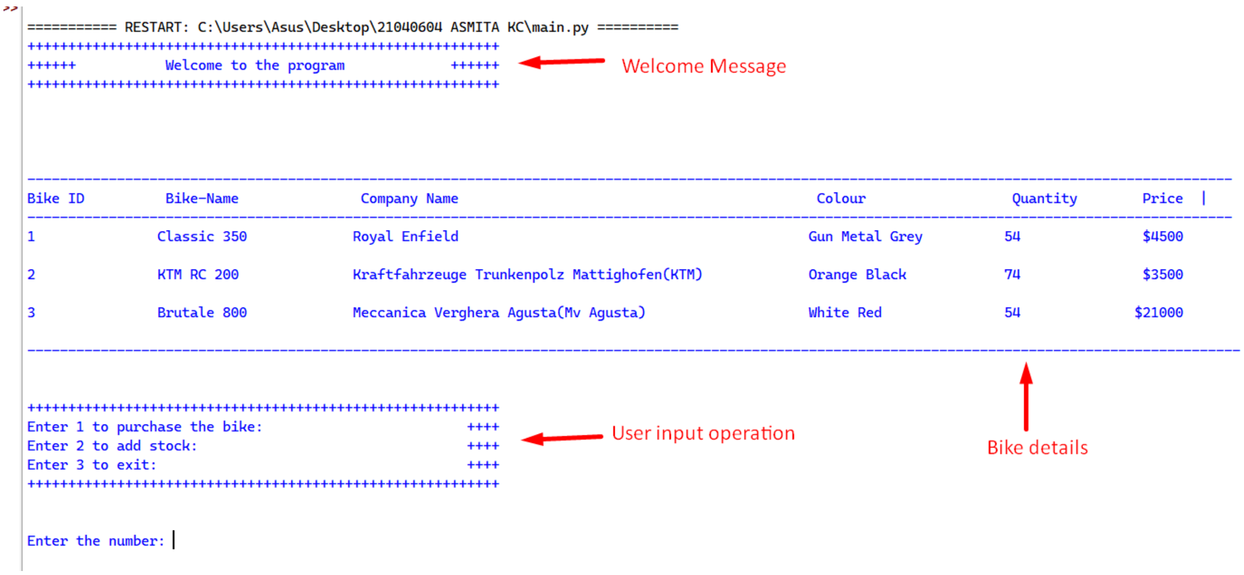
```
{1, 2, 3, 4}          Sets cannot have duplicate
{1, 2, 3, 4, 'Python'}  Added element in set
{1, 2, 3, 4}          Element removed from set
```

Figure 15 Example of Set

PROGRAM

This program is about a motorbike dealer maintaining its record or information in a text file. In this program, text file is a necessary asset as after each value input either a text file is generated or appended accordingly. This bike management system application is developed in Python software which helped a lot because of it’s high efficient data structure and easy to run ability. In this application user can add the bike or purchase one with the flow of the instruction given.

ENTER PROGRAM



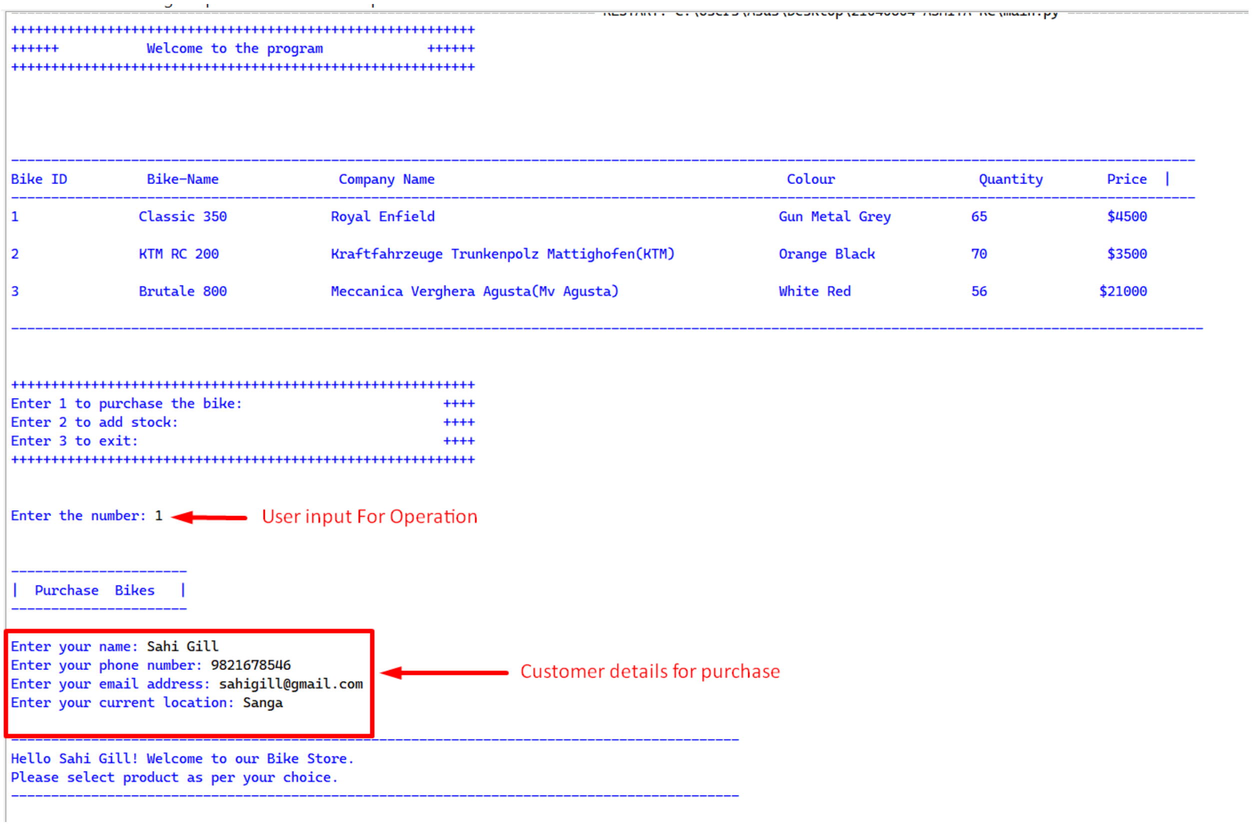
```
=========== RESTART: C:\Users\Asus\Desktop\21040604 ASMITA KC\main.py ==========
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++           Welcome to the program             ++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++




----------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name               Company Name                                            Colour                  Quantity        Price   |
----------------------------------------------------------------------------------------------------------------------------------------------------
1               Classic 350             Royal Enfield                                           Gun Metal Grey          54              $4500

2               KTM RC 200              Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)             Orange Black            74              $3500

3               Brutale 800             Meccanica Verghera Agusta(Mv Agusta)                    White Red               54             $21000

-----------------------------------------------------------------------------------------------------------------------------------------------------


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                         ++++
Enter 2 to add stock:                                 ++++
Enter 3 to exit:                                      ++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Enter the number: 
```

Figure 16 Program enter

When you enter or run the program welcome message appears along with the details of the bike which is possible because of the function to read the text file. Details of the bikes are extracted from the text file “bike.txt” and some options are given below. The given options include purchase bike, add bike and exit. The programs asks input to carry out the operation.

PROGRAM TO PURCHASE BIKE

Valid input to purchase bike



```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++           Welcome to the program             ++++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++




-----------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name               Company Name                                           Colour                  Quantity        Price  |
-----------------------------------------------------------------------------------------------------------------------------------------------------
1               Classic 350             Royal Enfield                                          Gun Metal Grey          65              $4500

2               KTM RC 200              Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)            Orange Black            70              $3500

3               Brutale 800             Meccanica Verghera Agusta(Mv Agusta)                   White Red               56              $21000

------------------------------------------------------------------------------------------------------------------------------------------------------


+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                         ++++
Enter 2 to add stock:                                 ++++
Enter 3 to exit:                                      ++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Enter the number: 1


----------------------
|  Purchase  Bikes   |
----------------------

Enter your name: Sahi Gill
Enter your phone number: 9821678546
Enter your email address: sahigill@gmail.com
Enter your current location: Sanga
------------------------------------------------------------------------------------------
Hello Sahi Gill! Welcome to our Bike Store.
Please select product as per your choice.
------------------------------------------------------------------------------------------
```

User input For Operation

Customer details for purchase

Figure 17 Input to purchase bike

When user press 1 in the user input field then the program for purchasing bike starts. To purchase anything you need to provide your valid information to store. In same way, this program also asks users to input their details to know who is purchasing and from where. After the input taken is completed, a formal welcome message appears to greet the user and provide details and further instruction on which bike to purchase.

Bike ID and quantity Validation

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                                    Colour                Quantity        Price  |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
1                Classic 350            Royal Enfield                                                   Gun Metal Grey        65              $4500

2                KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)                     Orange Black          70              $3500

3                Brutale 800            Meccanica Verghera Agusta(Mv Agusta)                            White Red             56              $21000
-----------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                                                                            Quantity before

Enter the ID of the bike you want: 3        Valid Bike ID


Enter the quantity you want to purchase: 11        Valid Quantity


-----------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                                    Colour                Quantity        Price  |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
1                Classic 350            Royal Enfield                                                   Gun Metal Grey        65              $4500

2                KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)                     Orange Black          70              $3500

3                Brutale 800            Meccanica Verghera Agusta(Mv Agusta)                            White Red             45              $21000
-----------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                                                                            Quantity After

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Congratulations!! The bike has been purchased. The details of the buyers are below:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 18 Quantity Validation

When user inputs valid bike id, it asks for quantity to purchase too. When user purchases the valid amount i.e. available in the stock, the quantity in the available stocks gets updated/reduced by the amount that was purchased and the updated stock gets displayed in the prompt.

Buyer's Details

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Congratulations!! The bike has been purchased. The details of the buyers are below:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Full Name:  Sahi Gill
Phone Number:  9821678546
Email-address:  sahigill@gmail.com
Current Location:  Sanga
Bike Name:  Brutale 800
Bike's Company Name:  Meccanica Verghera Agusta(Mv Agusta)
Bike Color:  White Red
Quantity of Bike:  11
Price of a single bike:  21000
Total cost of the bike:  231000
Date:  05/12/2022, 19:46:26
Do you want to purchase another bike? Y/N 1
++++++++++++++++++++++++++++++++++++++++++++

 Invalid Input!!! Please enter Y or N
++++++++++++++++++++++++++++++++++++++++++++
Do you want to purchase another bike? Y/N y

--------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                        Colour                 Quantity        Price  |
--------------------------------------------------------------------------------------------------------------------------------------------------------
1               Classic 350            Royal Enfield                                       Gun Metal Grey         65              $4500

2               KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)         Orange Black           70              $3500
```

Full Details of customer

Invalid Input

Invalid Input Message

Yes to purchase more

Program runs

Figure 19 Details of customers

When all valid inputs are given, the bike is successfully purchased and a message is displayed congratulating the user for successful purchase of the bike. Full details of the user is also displayed which was entered before along with the bike name, price, quantity, purchased date along with time. After the successful purchase of the bike, a question is raised if the user want to purchase more bike or not and is give two options Y as yes and N as no. If anything except of y or n is pressed then the loop runs until the satisfied input is given. If user enter y then the program continues from validating bike id.

Purchase again

```
Enter the ID of the bike you want: 1          <- Valid ID

Enter the quantity you want to purchase: 10   <- Valid quantity

-------------------------------------------------------------------------------------------------------------------------------
Bike ID        Bike-Name          Company Name                                    Colour              Quantity      Price  |
-------------------------------------------------------------------------------------------------------------------------------
1              Classic 350        Royal Enfield                                   Gun Metal Grey      55            $4500

2              KTM RC 200         Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)     Orange Black        70            $3500

3              Brutale 800        Meccanica Verghera Agusta(Mv Agusta)            White Red           45            $21000
-------------------------------------------------------------------------------------------------------------------------------

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Congratulations!! The bike has been purchased. The details of the buyers are below:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Full Name:  Sahi Gill
Phone Number:  9821678546
Email-address:  sahigill@gmail.com
Current Location:  Sanga
Bike Name:  Brutale 800
Bike's Company Name:  Meccanica Verghera Agusta(Mv Agusta)
Bike Color:  White Red
Quantity of Bike:  11
Price of a single bike:  21000
Total cost of the bike:  231000
Date:  05/12/2022, 19:46:26
Do you want to purchase another bike? Y/N n
```

Full Details again

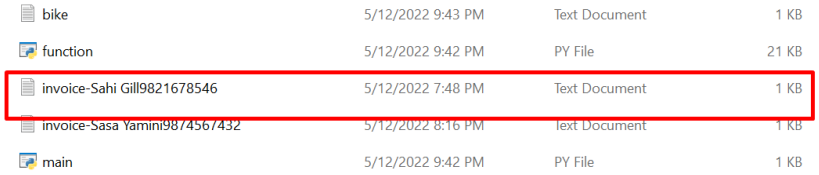
Figure 20 Purchase bike again

After yes is entered, the program runs again to take bike id and the wanted quantity. After valid inputs again a message congratulating along with the full details of the customer and the bikes is displayed. Again it asks if user want to purchase more and if no is entered in the program then the loop terminates saying The bikes were purchased and returns to the start of the program to enter the 3 options.

```
Do you want to purchase another bike? Y/N n   <- No any other purchase

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Thank you!! Your selected bikes were purchased successfully.   <- Loop end
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                                                Loop Continues until 3 entered

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                              ++++
Enter 2 to add stock:                                      ++++
Enter 3 to exit:                                           ++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Enter the number:
```

Figure 21 Loop after no entered

Bill generation



bike	5/12/2022 9:43 PM	Text Document	1 KB
function	5/12/2022 9:42 PM	PY File	21 KB
invoice-Sahi Gill9821678546	5/12/2022 7:48 PM	Text Document	1 KB
invoice-Sasa Yamini9874567432	5/12/2022 8:16 PM	Text Document	1 KB
main	5/12/2022 9:42 PM	PY File	1 KB

Figure 22 Creation of text file


```
                    Bike Management System
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Invoice: 2022-5-12-19-48-0          Date: 2022-5-12

                        Time: 19:48:0

Name of Customer: Sahi Gill
Phone Number: 9821678546
Email-address: sahigill@gmail.com
Current Location: Sanga


Bikes Purchased:
-------------------------------------------------------
Bike Name: Brutale 800
Company Name: Meccanica Verghera Agusta(Mv Agusta)
Bike Color: White Red
Quantity: 11
Price of Single Bike: $21000
Total Amount: $231000
-------------------------------------------------------
Bike Name: Classic 350
Company Name: Royal Enfield
Bike Color: Gun Metal Grey
Quantity: 10
Price of Single Bike:  $4500
Total Amount: $45000
-------------------------------------------------------

Grand Total: $276000
```

Figure 23 Purchase bill

After successful purchase of the bike, when user inputs no in the prompt, a bill is generated which includes the detail of the customer, bike purchased along with the amount quantity and grand total. The bill generation with append was possible due to the list and dictionary which made this an easy task to retrieve each information from the use by storing it.

PROGRAM TO ADD BIKE IN STOCK

Valid input to add bike

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                        ++++
Enter 2 to add stock:                                ++++
Enter 3 to exit:                                     ++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Enter the number: 2        Valid Input


------------------
|   Add  Stock   |
------------------

Enter your company name: Sasa Yamini
Enter company phone number: 9874567432
Enter company email address: sasaya@gmail.com
Enter the company current location: Saptari

Enter the name of the Shipping Company: ASA Traders        Company Detail input

Enter the shipping Cost: 34a

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Price can only contain numbers and $ (dollar) sign. Please try again!
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Enter the shipping Cost: $340

-------------------------------------------------------------------------------------------
Hello Sasa Yamini! Welcome to our Bike Store.
Please select product as per your choice.
-------------------------------------------------------------------------------------------


-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                                    Colour                  Quantity        Price  |
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
1               Classic 350            Royal Enfield                                                   Gun Metal Grey          54              $4500
```

Figure 24 Input to add bike stock

When user press 2 in the user input field then the program for adding bike to stock starts. To add anything you need to provide your valid information to store. In same way, this program also asks users to input their company details to know who is updating the stock along with the shipping company, shipping cost and from where. After the input taken is completed, a formal welcome message appears to greet the user and provide details and further instruction on which bike to add.

Bike ID and quantity Validation

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                          Colour                Quantity        Price  |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
1               Classic 350            Royal Enfield                                         Gun Metal Grey         54              $4500

2               KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)           Orange Black           70              $3500

3               Brutale 800            Meccanica Verghera Agusta(Mv Agusta)                  White Red              45             $21000

-----------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                                                                   Quantity before

Enter the ID of the bike you want: 2      Valid Bike ID

Enter the quantity you want to add: 4     Valid quantity


-----------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                          Colour                Quantity        Price  |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
1               Classic 350            Royal Enfield                                         Gun Metal Grey         54              $4500

2               KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)           Orange Black           74              $3500

3               Brutale 800            Meccanica Verghera Agusta(Mv Agusta)                  White Red              45             $21000

-----------------------------------------------------------------------------------------------------------------------------------------------------------
                                                                                                                   Quantity after

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The bike has been added to the stock. Following are the details of the company:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 25 Quantity Validation

When user inputs valid bike id, it asks for quantity to purchase too. When user adds the valid amount i.e. non negative quantity, the quantity in the available stocks gets updated/added by the amount that was added and the updated stock gets displayed in the prompt along with in the text file.

Company's Details

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The bike has been added to the stock. Following are the details of the company:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Company Name:  Sasa Yamini
Company's Phone Number:  9874567432
Company's Email-address:  sasaya@gmail.com
 Company's Current Location:  Saptari
Bike Name:  KTM RC 200
Bike's Company Name:  Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)
Bike Color:  Orange Black
Quantity of Bike:  4
Price of a single bike:   3500
Total cost of the bike:  14000
Name of Shipping Company:  ASA Traders
Shipping Cost:  $340
Date:  05/12/2022, 20:14:50
Do you want to add another bike? Y/N y

--------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                                    Colour                 Quantity        Price
--------------------------------------------------------------------------------------------------------------------------------------------------------
1               Classic 350            Royal Enfield                                                   Gun Metal Grey         54              $4500
```

Details of add

YES to add more

Figure 26 Details of company

When all valid inputs are given, the bike is successfully added and a message is displayed informing the user for successful addition of the bike. Full details of the user is also displayed which was entered before along with the bike name, price, quantity, added date along with time. After the successful addition of the bike, a question is raised if the user want to add more bike or not and is give two options Y as yes and N as no. If anything except of y or n is pressed then the loop runs until the satisfied input is given. If user enter y then the program continues from validating bike id.

Add stock again

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The bike has been added to the stock. Following are the details of the company:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Company Name:  Sasa Yamini
Company's Phone Number:  9874567432
Company's Email-address:  sasaya@gmail.com
 Company's Current Location:  Saptari
Bike Name:  KTM RC 200
Bike's Company Name:  Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)
Bike Color:  Orange Black
Quantity of Bike:  4
Price of a single bike:   3500
Total cost of the bike:  14000
Name of Shipping Company:  ASA Traders
Shipping Cost:  $340
Date:  05/12/2022, 20:14:50
Do you want to add another bike? Y/N n

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Thank you!! Your selected bikes were added successfully.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++



++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                        ++++
Enter 2 to add stock:                                ++++
Enter 3 to exit:                                     ++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Enter the number:
```

Full Details

No to end

Loop runs till 3 entered

Figure 27 Add bike again

After yes is entered, the program runs again to take bike id and the wanted quantity. After valid inputs again a message informing addition along with the full details of the company and the bikes is displayed. Again it asks if user wants to add more and if no is entered in the program then the loop terminates saying The bikes were added and returns to the start of the program to enter the 3 options.

ADD BILL

bike	5/12/2022 9:43 PM	Text Document	1 KB
function	5/12/2022 9:42 PM	PY File	21 KB
invoice-Sahi Gill9821678546	5/12/2022 7:48 PM	Text Document	1 KB
invoice-Sasa Yamini9874567432	5/12/2022 8:16 PM	Text Document	1 KB
main	5/12/2022 9:42 PM	PY File	1 KB

Figure 28 Creation of text file of add bike stock

invoice-Sasa Yamini9874567432 - Notepad

File Edit View

Bill ID with number

```
                    Bike Management System
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Invoice: 2022-5-12-20-16-26          Date: 2022-5-12

                              Time: 20:16:26

Name of Customer: Sasa Yamini
Phone Number: 9874567432
Email-address: sasaya@gmail.com
Current Location: Saptari


Bikes Added:
-------------------------------------------------------
Bike Name: KTM RC 200
Company Name: Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)
Bike Color: Orange Black
Quantity: 4
Price of Single Bike:  $3500
Total Amount: $14000
Shipping Company: ASA Traders
Shipping Cost: $340
-------------------------------------------------------
Bike Name: Brutale 800
Company Name: Meccanica Verghera Agusta(Mv Agusta)
Bike Color: White Red
Quantity: 9
Price of Single Bike: $21000
Total Amount: $189000
Shipping Company: ASA Traders
Shipping Cost: $340
-------------------------------------------------------

Grand Total: $203680
```

Total Price with Shipping Cost included

Figure 29 Bike add bill

After successful addition of the bike, a bill is generated which includes the detail of the company, bike added along with the amount, quantity, shipping company, shipping cost and grand total. The bill generation with append was possible due to the list and dictionary which made this an easy task to retrieve each information from the use by storing it.

EXIT PROGRAM

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                          ++++
Enter 2 to add stock:                                  ++++
Enter 3 to exit:                                       ++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Enter the number: 3        3 to exit the system


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Thank you for using our system
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
>>>
```

Figure 30 Exit Program

When user wishes to terminate or exit the system, entering 3 will satisfy the loop to end which breaks the program and terminates it.

IMPLEMENTATION OF TRY EXCEPT

Try except is used in exception handling which brings out the best in a program and makes you much safe from any error to arise. In this program too, try except is used. As shown in the below picture, Invalid input is displayed when you enter non-existing value and displays provide integer input when string is entered in case of integer. The program only steps ahead when the condition kept is satisfied, otherwise, it asks user to follow the rules given.

```
-------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                          Colour                Quantity        Price  |
-------------------------------------------------------------------------------------------------------------------------------------------------------
1                Classic 350            Royal Enfield                                         Gun Metal Grey        65              $4500

2                KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)           Orange Black          70              $3500

3                Brutale 800            Meccanica Verghera Agusta(Mv Agusta)                  White Red             56             $21000

-------------------------------------------------------------------------------------------------------------------------------------------------------


++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                        ++++
Enter 2 to add stock:                                ++++
Enter 3 to exit:                                     ++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Enter the number: 4        <- Non-existing value

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Invalid input!!!                                          <- Invalid input
Please provide value as 1, 2 or 3.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Enter the number: A        <- String input

++++++++++++++++++++++++++++++++++++++++
Please provide Integer Value!!!!           <- Only Integer
++++++++++++++++++++++++++++++++++++++++

Enter the number: 1        <- Valid Input

----------------------
|  Purchase  Bikes   |     <- Program runs
----------------------
```

Figure 31 Try-except in bike id

```
Enter your name: 123                                   Integer input in name

++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Name only contains alphabets. Please try again!        Invalid message
++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Enter your name: Sahi Gill                             Takes string input
Enter your phone number: 2asb51                        String input in number

++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Phone number only contain numbers. Please try again!   Invalid message
++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Enter your phone number: 9876124356                    Takes integer input
Enter your email address: gillsahi@gmail.com
Enter your current location: Sakhu

----------------------------------------------------------------------------------------
Hello Sahi Gill! Welcome to our Bike Store.
Please select product as per your choice.
----------------------------------------------------------------------------------------
```

Figure 32 Try-except in name and number

When user inputs integer value except of string then the exception occurs which throws message name can only contain alphabets. When user tries to input string on number error occurs which is handled by ValueError in python. Hence, phone numbers are taken as integer and returned as string. Error is handled when user input mixed data types rather than integer with $ sign.

```
Enter the shipping Cost: 34a                           Invalid input

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Price can only contain numbers and $ (dollar) sign. Please try again!
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Enter the shipping Cost: $340                          Input accepted

--------------------------------------------------------------------------------
Hello Sasa Yamini! Welcome to our Bike Store.
```

Figure 33 Exception in price

TESTING

TEST 1: IMPLEMENTATION OF TRY EXCEPT

TEST 1.1: IMPLEMENTATION OF TRY EXCEPT IN NAME

Test No:	1.1
Objective:	Showing Implementation of try, except in name
Action:	Provide invalid input in name i.e., integer in string
Expected Result:	Exception throwing message “Name only contains alphabets. Please try again!”
Actual Result:	Exception was thrown with the message “Name only contains alphabets. Please try again!”
Conclusion	The test was successful.

Table 1 To Test Use of Try Except in Name Input

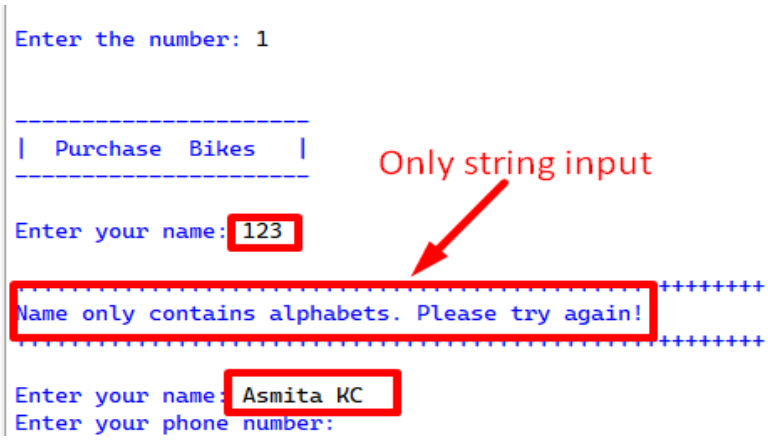


Figure 34 Use of try-except in name input

TEST 1.2: IMPLEMENTATION OF TRY EXCEPT IN NAME

Test No:	1.2
Objective:	Showing Implementation of try, except
Action:	Provide invalid input in Bike ID i.e., string on integer
Expected Result:	Exception throwing message “Please enter integer value from the table!!!”
Actual Result:	Exception was thrown with the message “Please enter integer value from the table!!!”
Conclusion	The test was successful.

Table 2 To Test Use of Try Except in Bike ID input

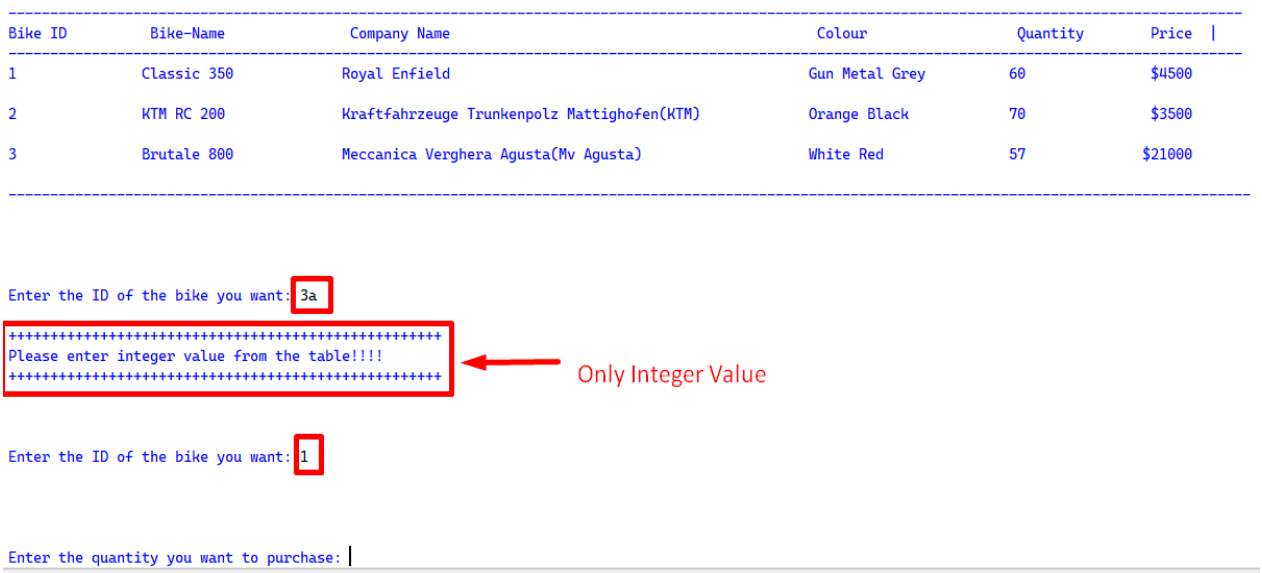


Figure 35 Use of try-except in Bike ID input

TEST 2: SELECTION ADDING BIKES IN STOCK AND SELLING OF BIKES

TEST 2.1: SELECTION SELLING OF BIKES

Test No:	2.1
Objective:	Selection selling of bikes.
Action:	Provide negative value on the quantity and invalid bike ID
Expected Result:	Exception throwing message provide valid bike id and valid quantity.
Actual Result:	Exception was thrown with the message provide valid bike id and valid quantity.
Conclusion	The test was successful.

Table 3 To Test Selection Selling of Bikes

Figure 36 Selection Selling of Bikes In Program

TEST 2.2: SELECTION ADDING OF BIKES

Test No:	2.2
Objective:	Selection Adding of bikes.
Action:	Provide negative value on the quantity and invalid bike ID
Expected Result:	Exception throwing message provide valid bike id and valid quantity.
Actual Result:	Exception was thrown with the message provide valid bike id and valid quantity.
Conclusion	The test was successful.

Table 4 To Test Selection Adding of Bikes

```
Enter the ID of the bike you want: 5        <- Invalid Bike ID

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Please provide a valid Bike ID !!!          <- Invalid Bike ID message display
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

-------------------------------------------------------------------------------------------------------------------------------------------
Bike ID     Bike-Name          Company Name                                    Colour            Quantity     Price  |
-------------------------------------------------------------------------------------------------------------------------------------------
1           Classic 350        Royal Enfield                                   Gun Metal Grey    58           $4500
2           KTM RC 200         Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)     Orange Black      70           $3500
3           Brutale 800        Meccanica Verghera Agusta(Mv Agusta)            White Red         57           $21000
-------------------------------------------------------------------------------------------------------------------------------------------

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter the ID of the bike you want: 3        <- Valid Bike ID
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Enter the quantity you want to add: -80     <- Negative quantity addition

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Please provide a valid Quantity ID !!!      <- Invalid Quantity message display
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Enter the quantity you want to add: 2
```

Figure 37 Selection Adding of Bikes in Program

TEST 3: FILE GENERATION OF SELLING BIKES

Test No:	3
Objective:	File generation of selling bikes.
Action:	Sell bike with valid input.
Expected Result:	Sell of bike with text file generation.
Actual Result:	Bike was sold along with the text file as bill.
Conclusion	The test was successful.

Table 5 To Test File Generation of Selling Bikes

```
=========== RESTART: C:\Users\Asus\Desktop\21040604 ASMITA KC\main.py ==========
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++          Welcome to the program            ++++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Welcome Message

Bike Display

Bike ID	Bike-Name	Company Name	Colour	Quantity	Price
1	Classic 350	Royal Enfield	Gun Metal Grey	58	$4500
2	KTM RC 200	Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)	Orange Black	68	$3500
3	Brutale 800	Meccanica Verghera Agusta(Mv Agusta)	White Red	58	$21000

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                          ++++
Enter 2 to add stock:                                  ++++
Enter 3 to exit:                                       ++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Option Display

```
Enter the number: 1
```

Ask user for operation

```
----------------------
|  Purchase  Bikes   |
----------------------

Enter your name: Asmita KC
Enter your phone number: 9876325631
Enter your email address: asmita@gmail.com
Enter your current location: Kathmandu
```

Input User details

```
-------------------------------------------------------------------------------------
Hello Asmita KC! Welcome to our Bike Store.
Please select product as per your choice.
-------------------------------------------------------------------------------------
```

Formal welcome message to customer

Figure 38 Input of the details to purchase

Bike ID	Bike-Name	Company Name	Colour	Quantity	Price
1	Classic 350	Royal Enfield	Gun Metal Grey	58	$4500
2	KTM RC 200	Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)	Orange Black	68	$3500
3	Brutale 800	Meccanica Verghera Agusta(Mv Agusta)	White Red	58	$21000

Quantity before

```
Enter the ID of the bike you want: 3
```

Input Bike ID

```
Enter the quantity you want to purchase 2
```

Quantity to purchase

Bike ID	Bike-Name	Company Name	Colour	Quantity	Price
1	Classic 350	Royal Enfield	Gun Metal Grey	58	$4500
2	KTM RC 200	Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)	Orange Black	68	$3500
3	Brutale 800	Meccanica Verghera Agusta(Mv Agusta)	White Red	56	$21000

Quantity after sell

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Congratulations!! The bike has been purchased. The details of the buyers are below:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Full Name:  Asmita KC
Phone Number:  9876325631
Email-address:  asmita@gmail.com
Current Location:  Kathmandu
Bike Name:  Brutale 800
Bike's Company Name:  Meccanica Verghera Agusta(Mv Agusta)
Bike Color:  White Red
Quantity of Bike:  2
Price of a single bike:  21000
Total cost of the bike:  42000
Date:  05/12/2022, 09:34:54
Do you want to purchase another bike? Y/N n
```

Full Detail

If user wants to purchase more

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Thank you!! Your selected bikes were purchased successfully.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Thank you message

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                        ++++
Enter 2 to add stock:                                ++++
Enter 3 to exit:                                     ++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Enter the number:
```

Loop continues until 3 entered

Figure 39 Display of details with quantity variation after purchase

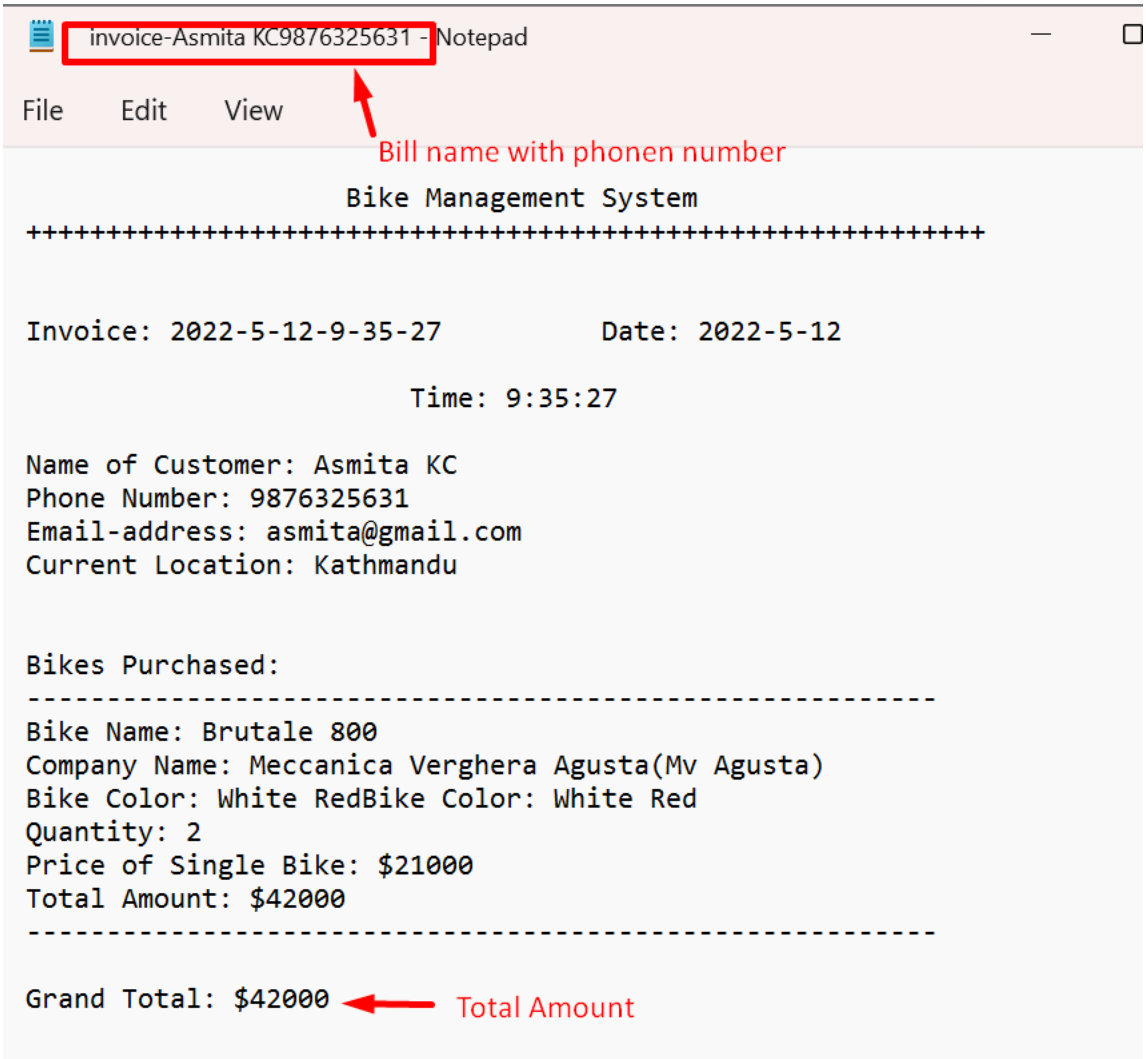


Figure 40 Generation of purchase bill in text file

TEST 4: FILE GENERATION OF ADDING BIKES IN STOCK

Test No:	4
Objective:	File generation of adding bikes.
Action:	Add bike with valid input.
Expected Result:	Addition of bike with text file generation.
Actual Result:	Bike was added along with the text file as bill.
Conclusion	The test was successful.

Table 6 To Test File Generation of Adding Bikes in Stock

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++          Welcome to the program            ++++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Welcome message

Display Bike

```
-------------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                          Colour                Quantity        Price  |
-------------------------------------------------------------------------------------------------------------------------------------------------------------
1               Classic 350            Royal Enfield                                         Gun Metal Grey        62              $4500

2               KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)           Orange Black          72              $3500

3               Brutale 800            Meccanica Verghera Agusta(Mv Agusta)                  White Red             56             $21000

-------------------------------------------------------------------------------------------------------------------------------------------------------------


+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                        ++++
Enter 2 to add stock:                                ++++
Enter 3 to exit:                                     ++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Display Options

```
Enter the number: 2
```

Input for operation

```
------------------
|   Add  Stock   |
------------------

Enter your company name: ASA Traders
Enter company phone number: 989987456
Enter company email address: asat@gmail.com
Enter the company current location: Baneshwor

Enter the name of the Shipping Company: SAYU Bikes

Enter the shipping Cost: $500
```

Company Details

```
-------------------------------------------------------------------------------------------------
Hello ASA Traders! Welcome to our Bike Store.
Please select product as per your choice.
-------------------------------------------------------------------------------------------------
```

Formal welcome message

Figure 41 Input of the details to purchase

```
-----------------------------------------------------------------------------------------------------------------------------------
Bike ID        Bike-Name          Company Name                                    Colour              Quantity      Price  |
-----------------------------------------------------------------------------------------------------------------------------------
1              Classic 350        Royal Enfield                                   Gun Metal Grey      62            $4500

2              KTM RC 200         Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)     Orange Black        72            $3500

3              Brutale 800        Meccanica Verghera Agusta(Mv Agusta)            White Red           5             $21000

-----------------------------------------------------------------------------------------------------------------------------------
```

Quantity before

```
Enter the ID of the bike you want: 2
```

Bike ID

```
Enter the quantity you want to add: 3
```

Quantity to add

```
-----------------------------------------------------------------------------------------------------------------------------------
Bike ID        Bike-Name          Company Name                                    Colour              Quantity      Price  |
-----------------------------------------------------------------------------------------------------------------------------------
1              Classic 350        Royal Enfield                                   Gun Metal Grey      62            $4500

2              KTM RC 200         Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)     Orange Black        75            $3500

3              Brutale 800        Meccanica Verghera Agusta(Mv Agusta)            White Red           8             $21000

-----------------------------------------------------------------------------------------------------------------------------------
```

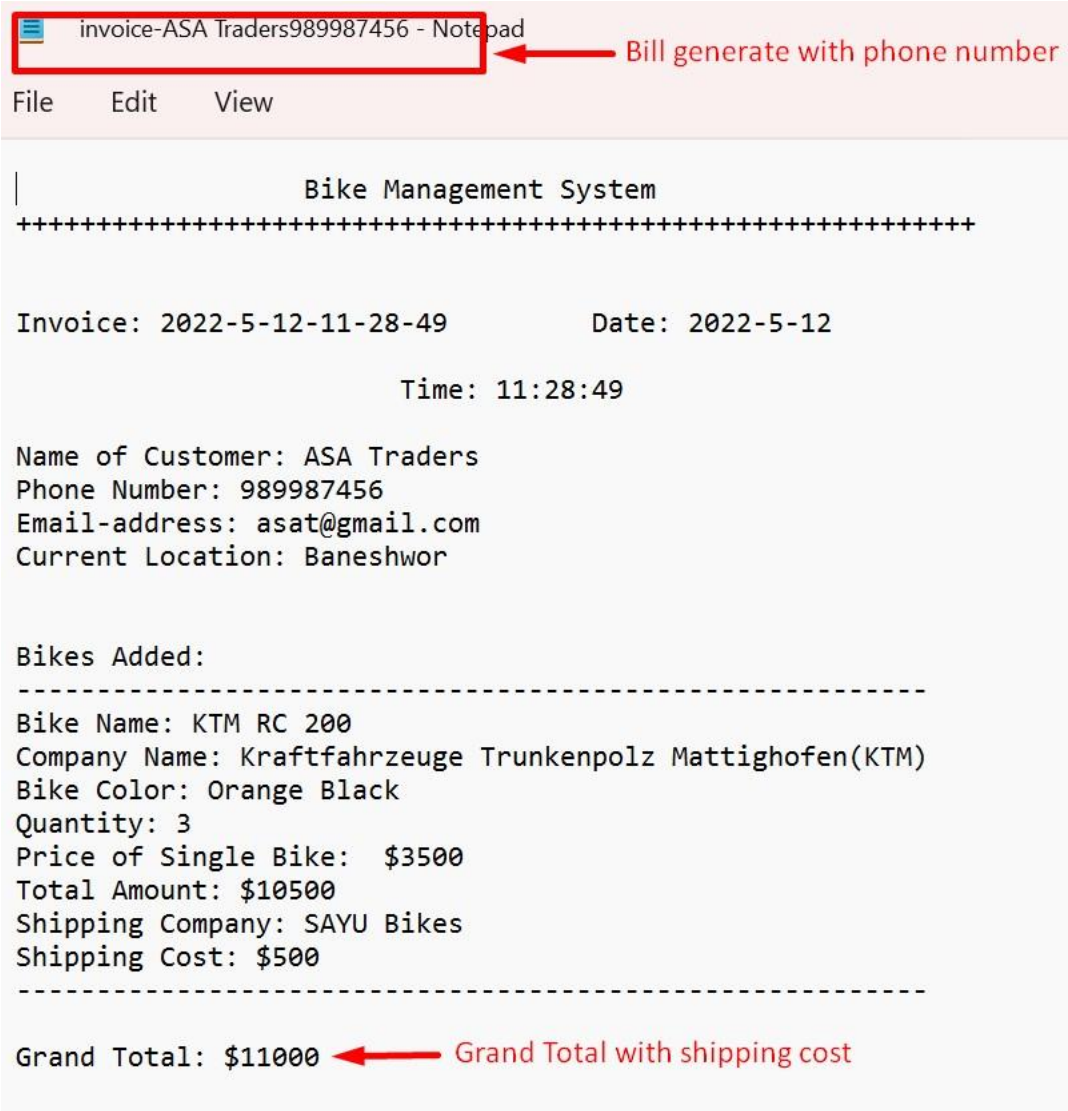
Quantity After

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The bike has been added to the stock. Following are the details of the company:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Company Name:  ASA Traders
Company's Phone Number:  989987456
Company's Email-address:  asat@gmail.com
 Company's Current Location:  Baneshwor
Bike Name:  KTM RC 200
Bike's Company Name:  Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)
Bike Color:  Orange Black
Quantity of Bike:  3
Price of a single bike:   3500
Total cost of the bike:  10500
Name of Shipping Company:  SAYU Bikes
Shipping Cost:  $500
Date:  05/12/2022, 11:27:26
Do you want to add another bike? Y/N n

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 Thank you!! Your selected bikes were added successfully.
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Enter 1 to purchase the bike:                        ++++
Enter 2 to add stock:                                ++++
Enter 3 to exit:                                     ++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Enter the number:
```

Full Details

Loop runs until entered 3

Figure 42 Display of details with quantity variation after purchase




```
                    Bike Management System
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Invoice: 2022-5-12-11-28-49         Date: 2022-5-12

                         Time: 11:28:49

Name of Customer: ASA Traders
Phone Number: 989987456
Email-address: asat@gmail.com
Current Location: Baneshwor


Bikes Added:
-------------------------------------------------------
Bike Name: KTM RC 200
Company Name: Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)
Bike Color: Orange Black
Quantity: 3
Price of Single Bike:  $3500
Total Amount: $10500
Shipping Company: SAYU Bikes
Shipping Cost: $500
-------------------------------------------------------

Grand Total: $11000
```

Grand Total with shipping cost

Figure 43 Generation of added bill in text file

TEST 5: SHOW THE UPDATE IN STOCK OF BIKE

TEST 5.1: QUANTITY UPDATE AFTER SELLING BIKE

Test No:	4
Objective:	Quantity Update while selling bike
Action:	Purchase bike with valid input.
Expected Result:	Quantity deduction with update in table
Actual Result:	Quantity deducted in table.
Conclusion	The test was successful.

Table 7 To Test stock update in bike while selling

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                          Colour                 Quantity        Price  |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
1                Classic 350            Royal Enfield                                         Gun Metal Grey         62              $4500

2                KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)           Orange Black           75              $3500

3                Brutale 800            Meccanica Verghera Agusta(Mv Agusta)                  White Red              56              $21000

-----------------------------------------------------------------------------------------------------------------------------------------------------------
```

Quantity before purchase

```
Enter the ID of the bike you want: 2


Enter the quantity you want to purchase: 5
```

Quantity Purchased

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                          Colour                 Quantity        Price  |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
1                Classic 350            Royal Enfield                                         Gun Metal Grey         62              $4500

2                KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)           Orange Black           70              $3500

3                Brutale 800            Meccanica Verghera Agusta(Mv Agusta)                  White Red              56              $21000

-----------------------------------------------------------------------------------------------------------------------------------------------------------
```

Quantity deducted after purchasing

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Congratulations!! The bike has been purchased. The details of the buyers are below:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 44 Quantity update after purchase

TEST 5.2: QUANTITY UPDATE AFTER ADDING BIKE

Test No:	4
Objective:	Quantity Update while selling bike
Action:	Purchase bike with valid input.
Expected Result:	Quantity deduction with update in table
Actual Result:	Quantity deducted in table.
Conclusion	The test was successful.

Figure 45 To Test stock update in bike while adding

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                          Colour                 Quantity        Price  |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
1               Classic 350            Royal Enfield                                         Gun Metal Grey          62              $4500

2               KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)           Orange Black            70              $3500

3               Brutale 800            Meccanica Verghera Agusta(Mv Agusta)                  White Red               56             $21000

-----------------------------------------------------------------------------------------------------------------------------------------------------------
```

Quantity before add

```
Enter the ID of the bike you want: 1

Enter the quantity you want to add: 3
```

ADD BIKE QUANTITY

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
Bike ID          Bike-Name              Company Name                                          Colour                 Quantity        Price  |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
1               Classic 350            Royal Enfield                                         Gun Metal Grey          65              $4500

2               KTM RC 200             Kraftfahrzeuge Trunkenpolz Mattighofen(KTM)           Orange Black            70              $3500

3               Brutale 800            Meccanica Verghera Agusta(Mv Agusta)                  White Red               56             $21000

-----------------------------------------------------------------------------------------------------------------------------------------------------------
```

Quantity after add

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The bike has been added to the stock. Following are the details of the company:
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 46 Quantity update after adding

CONCLUSION

The coursework focuses on acquiring new skills and using those skills in a real-word context. This is an individual assessment which weighs more than 60% for this semester and is required to do it in Python application. This coursework was quite benificial in clearing out any misunderstandings, logics and it conveys all the things we have learned till now in the module.

During the compilation the program, many mistakes were encountered and many errors got caught. The concept of 2D list and while loop played main character during the coursework along with the dictionary. While loop is the only one making our program run until the condition we gave is satisfied. This helped a lot in understanding how really a Python work in it's environment. The implementation try-except was done,which made the work perfect, as it handles the file and every errors encountered. The readability function of python and the perfect and simple use of data structure made this coursework much of fun and exciting. It was a great pleasure to work in python application encountering various problems alongside discussing with the tutors and lecturers.

As this is the first time I am familiar with the Python course, it was really challenging for me to discover about algorithm and flowchart along with the development of the program itself. It was hard to understand the concept and various difficulties were raised and encountered. Some errors went undetectable and un-noticed but with the help from the tutors, I dealt through the code part. And finally I've accomplished it with all the hardwork and joy. The accomplishment of the coursework has given insight on the knowledge I previously had. It has really aided in gaining sufficient information and creativity to complete my assignment, and all was made possible by my mentor, who assisted and directed me through many obstacles, and I am confident that this will be extremely beneficial in the future days.

REFERENCES

Real Python, 2020. *Common python data structures (guide).* s.l.:Real Python.

Elrawy, O., 2017. *The Integration of Buildings' Energy Simulation Tools (ESTs) with Intelligent Decision Support Systems (IDSS).* s.l.:s.n.

GeeksforGeeks, 2021. *GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/
[Accessed April 2022].

Kilonzi, F. M., 2020. *Data structures in python.* s.l.:DEV Community.

LucidChart, 2022. *What is a Flowchart.* s.l.:Lucid Chart.

Ritesh, R., 2021. *What is Python & Why Does it Matter in Software Development?.* s.l.:Net Solution.

Techopedia, 2021. *Algorithm.* s.l.:Techopedia.

TUTORIALINK, 2022. *Tutorialink.com.* [Online]
Available at: https://tutorialink.com/ds/basic-concepts-of-data-structure.ds
[Accessed April 2022].

UpGrad, 2020. *Data structures & algorithm in Python: Everything you need to know.* s.l.:upGrad blog.

Van, R., Guido, Drake, J. & Fred, L., 1995. *Python tutorial.* Amsterdam: Centrum voor Wiskunde en Informatica Amsterdam.

Writh, N., 1976. *Algorithms + data structures=programs.* Englewood Cliffs, N.J.: Englewood Cliffs, N.J. : Prentice-Hall.

APPENDICES

APPENDIX A: FUNCTION FILE

```
#importing date and time
import datetime
#the given below function displays the welcome message to the user
def welcome():
    print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
    print("++++++          Welcome to the program          ++++++")
    print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
    print("\n")


#the given function displays the bikes from the text file to the user
def display_bike():
    print("\n")
    print("-------------------------------------------------------------------------------------------------------------------------------------------")
    print("Bike ID \t Bike-Name \t\t Company Name \t\t\t\t\t\t Colour \t\t Quantity \t Price \t|")
    print("-------------------------------------------------------------------------------------------------------------------------------------------")
    try:
```

```
        #open text file in read mode
        file = open("bike.txt","r")
        a = 1
        for line in file:
            print(a,"\t\t"+line.replace(",","\t\t"))
            a= a+1
        print("-------------------------------------------------------------------------------------------------------
-----------------------------------------------")
        print("\n")
        file.close()
        #close file
    #check for exception
    except IOError:
        print("\nFile name misplaced!")


#the given function adds the bikes from the text file to the 2D list
def add_bike_2D_list():
    #open text file in read mode
    read_file = open("bike.txt","r")
    my_list = []
    for i in read_file:
```

```
        i = i.replace("\n","")
        my_list.append(i.split(","))
        #store bike details in 2D list
    return my_list



#the given below function displays options for the users in the system
def user_operation():
    print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
    print("Enter 1 to purchase the bike:                    ++++")
    print("Enter 2 to add stock:                          ++++")
    print("Enter 3 to exit:                             ++++")
    print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
    print("\n")


#the given below function displays bike added to stock  when user press 2
def add_bike_stock():
    print("\n")


print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
```

```
    print("The bike has been added to the stock. Following are the details of the
company:")

print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++")
    print("\n")


#the given below function validate the bike ID
def validating_bike_id():
    loop = True
    #loop runs
    while loop == True:
        try:
            print("\n")
            global valid_id
            #declare global to access it from other functions
            valid_id = int(input("Enter the ID of the bike you want: "))
            print("\n")
            #check if input is greater than 0 and less than the bike listed
            while valid_id<=0 or valid_id>len(add_bike_2D_list()):

print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
```

```
            print("Please provide a valid Bike ID !!!")

print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
            print("\n")
            display_bike()
            #display bike after invalid input
            print("\n")

print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
            valid_id = int(input("Enter the ID of the bike you want: "))  #ask to input again
after invalid input

print("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
            print("\n")
        return valid_id #return the id entered
        loop = False
        #loop terminates
      except ValueError:  #check for value error to catch exception
        print("\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
        print("Please enter integer value from the table!!!!")
        print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n")
```

```
#the given below function displays bike purchased from stock  when user press 1

def purchase_bike():

    print("\n")


print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++")

    print("Congratulations!! The bike has been purchased. The details of the buyers are
below:")


print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++")

    print("\n")




#the given below function displays invalid user input when the user gives invalid input

def invalid_user_input():

    print("\n")

    print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++")

    print("Invalid input!!!")

    print("Please provide value as 1, 2 or 3.")

    print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++")
```

```
    print("\n")


#the given below function displays system exit when the user press 3
def exit_system():
    print("\n")
    print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
    print("Thank you for using our system")
    print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")


#the given below function updates bike stock in text file
def update_stock(bike_list):
    #open text file in write mode
    file = open("bike.txt","w")
    for i in bike_list:
        file.write(str(i[0])+","+str(i[1])+","+str(i[2])+","+str(i[3])+","+str(i[4])+"\n")
        #update bike stock after purchasing bike
    file.close()
    #file closed


    display_bike()
    #print bike details
```

```
#the given below function updates bike stock in text file by adding quantity
def add_stock(bike_list):
    #open text file in write mode
    file = open("bike.txt","w")
    for i in bike_list:
        file.write(str(i[0])+","+str(i[1])+","+str(i[2])+","+str(i[3])+","+str(i[4])+"\n")
        #update bike stock after adding bike
    file.close()
    #file closed

    display_bike()
    #print bike details

#the given below function updates bike quantity from user
def final_add(user_input_bike_id, the_quantity):
    bike_list = add_bike_2D_list()
    bike_list[user_input_bike_id - 1][3] = int(bike_list[user_input_bike_id - 1][3]) +
the_quantity
    add_stock(bike_list)
```

```
#the given below function validates quantity input
def quantity_validation(the_bike_id):
    loop = True
    #loop runs
    while loop == True:
        #using of try except for Value Error
        try:
            bike_list = add_bike_2D_list()
            user_quantity = int(input("Enter the quantity you want to purchase: "))
            print("\n")
            while user_quantity <=0 or user_quantity>int(bike_list[the_bike_id - 1][3]):
                print("\n")

print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
                print("Please provide a valid Quantity ID !!!")

print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
                print("\n")
                user_quantity = int(input("Enter the quantity you want to purchase: "))
                print("\n")
            return user_quantity
```

```
            loop = False
        except ValueError:
            print("\n+++++++++++++++++++++++++++++++++++++++")
            print("Only Integer Values Please!!!!")
            print("+++++++++++++++++++++++++++++++++++++++\n")


#the given below function updates bike quantity after sell to user
def final_sell(user_input_bike_id, the_quantity):
    bike_list = add_bike_2D_list()
    bike_list[user_input_bike_id - 1][3] = int(bike_list[user_input_bike_id - 1][3]) -
the_quantity
    update_stock(bike_list)


#the given below function calculates total price
def total_price(user_input_bike_id, the_quantity):
    bike_list = add_bike_2D_list()
    total_price = int(bike_list[user_input_bike_id-1][4].replace("$",""))*the_quantity
    return total_price



#the given below function validates quantity input
```

```
def quantity_val(the_bike_id):
    loop = True
    while loop == True:
        try:
            global bike_list, id
            #declaring global so it can be accessed


            bike_list = add_bike_2D_list()
            id = the_bike_id
            user_quantity = int(input("Enter the quantity you want to add: "))
            print("\n")
            #checking if quantity input is greater than 0
            while user_quantity <=0:
                print("\n")

print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
                print("Please provide a valid Quantity ID !!!")

print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
                print("\n")
                user_quantity = int(input("Enter the quantity you want to add: "))
```

```
            print("\n")
        return user_quantity
        loop = False
        #loop breaks

    except ValueError:
        print("\n+++++++++++++++++++++++++++++++++++++++")
        print("Please Integer Value Only!!!!")
        print("+++++++++++++++++++++++++++++++++++++++\n")


#the given below function returns selected bike color to user
def color(user_input_bike_id):
    bike_list = add_bike_2D_list()
    bcolor = (bike_list[user_input_bike_id-1][2])
    print("Bike Color: ", bcolor)
    return bcolor


#the given below function returns selected bike name to user
def nameb(user_input_bike_id):
```

```
    bike_list = add_bike_2D_list()
    bname = (bike_list[user_input_bike_id-1][0])
    print("Bike Name: ", bname)
    return bname

#the given below function returns selected bike company name to user
def cname(user_input_bike_id):
    bike_list = add_bike_2D_list()
    cname = (bike_list[user_input_bike_id-1][1])
    print("Bike's Company Name: ", cname)
    return cname

#the given below function returns selected bike price to user
def price(user_input_bike_id):
    bike_list = add_bike_2D_list()
    bprice = (bike_list[user_input_bike_id-1][4].replace("$",""))
    print("Price of a single bike: ", bprice)
    return bprice

#the given below function checks if valid price is given
def validPrice(text):
```

```
    price = input(text)
    try:
        price = int(price.replace('$',''))
        #check input in integer and catch exception
    except:

print("\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
        print('Price can only contain numbers and $ (dollar) sign. Please try again!')

print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n")
        price = validPrice(text).replace('$','')
    return '$'+str(price)


#the given below function checks integer and string
def number(text):
    in_num = input(text)
    try:
        in_num = int(in_num)
        #check input in integer and catch exception
    except:
```

```
print("\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
        print('Phone number only contain numbers. Please try again!')

print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n")
        in_num = number(text)
    return str(in_num)


#the given below function checks name is in string
def check_name(text):
    in_name = input(text)
    st1 = in_name.replace(" ","").isalpha()
    try:
        if not st1:
            raise ValueError
            #raises valueerror if it contains any integer
    except ValueError:

print("\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
        print('Name only contains alphabets. Please try again!')

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

```
        in_name = check_name(text)
    return str(in_name)


#the given below function takes user details input
def user():
    global name, num, email, location, curr_date
    name = check_name("Enter your name: ") #call check name function from above
    num = number("Enter your phone number: ") #call check number function from above
    email = str(input("Enter your email address: "))
    location = str(input("Enter your current location: "))


    date = datetime.datetime.now()
    curr_date = date.strftime("%m/%d/%Y, %H:%M:%S")
    print("\n-----------------------------------------------------------------------------------------------")
    print("Hello " + name + "! Welcome to our Bike Store.\nPlease select product as per
your choice.")
    print("-----------------------------------------------------------------------------------------------\n")
    display_bike()


#the given below function prints user details
def puser():
```

```
    print("Full Name: ", name)

    print("Phone Number: ", num)

    print("Email-address: ", email)

    print("Current Location: ",location)

    nameb(the_bike_id)

    cname(the_bike_id)

    color(the_bike_id)

    print("Quantity of Bike: ", the_q)

    price(the_bike_id)

    print("Total cost of the bike: ", the_price)

    print("Date: ", curr_date)



#the given below function asks if user wants to purchase more

def user_ask():

    run = True

    #loop runs till False not returned

    while run == True:

        print("\n")

        ask = input("Do you want to purchase another bike? Y/N ")
```

```
        #if input is yes
        if ask.upper() == "Y":
            display_bike() #display bike details
            the_bike_id = validating_bike_id() #validate bike id
            the_q = quantity_validation(the_bike_id) #validate bike quantity
            final_sell(the_bike_id, the_q) #finalize the sell
            the_price = total_price(the_bike_id, the_q) #calculate price
            bike.update({the_bike_id:the_q}) #update bike list of user
            bill(the_bike_id, bike, name, num, email, location, curr_date) #append bill after
purchase again
            purchase_bike() #purchase message display
            puser() #display user details

        #if input is no
        elif ask.upper() == "N":

print("\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
            print("Thank you!! Your selected bikes were purchased successfully.")

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

```
            run = False
            #loop terminates


        #input except yes and no
        else:
            print("+++++++++++++++++++++++++++++++++++++++++++++")
            print("\n Invalid Input!!! Please enter Y or N ")
            print("+++++++++++++++++++++++++++++++++++++++++++++")


#the given below function takes company details input
def com_user():
    global comp_name, comp_number, comp_email, comp_location, current_date, ship, shipCost
    #ask for input
    comp_name = check_name("Enter your company name: ") #call check name function from above
    comp_number = number("Enter company phone number: ") #call check number function from above
    comp_email = input("Enter company email address: ")
    comp_location = input("Enter the company current location: ")
    date = datetime.datetime.now()
    current_date = date.strftime("%m/%d/%Y, %H:%M:%S")
```

```
    ship = input("\nEnter the name of the Shipping Company: ")

    shipCost = validPrice("\nEnter the shipping Cost: ") #call check valid price function
from above

    print("\n-----------------------------------------------------------------------------------------------")

    print("Hello " + comp_name + "! Welcome to our Bike Store.\nPlease select product
as per your choice.")

    print("-----------------------------------------------------------------------------------------------\n")

    display_bike()


#the given below function prints company details

def comp_user():

    print("Company Name: ", comp_name)

    print("Company's Phone Number: ", comp_number)

    print("Company's Email-address: ", comp_email)

    print(" Company's Current Location: ", comp_location)

    nameb(the_bike_id)

    cname(the_bike_id)

    color(the_bike_id)

    print("Quantity of Bike: ",the_q)

    price(the_bike_id)

    print("Total cost of the bike: ", the_price)
```

```
    print("Name of Shipping Company: ", ship)
    print("Shipping Cost: ", shipCost)
    print("Date: ", current_date)


#the given below function asks if user wants to add more
def comp_ask():
    run = True
    while run == True:
        print("\n")
        ask = input("Do you want to add another bike? Y/N ")
        #if input is yes
        if ask.upper() == "Y":
            display_bike() #displaying bike
            the_bike_id = validating_bike_id() #validate bike id
            the_q = quantity_val(the_bike_id) #validate quantity
            final_add(the_bike_id, the_q) #finalize add
            the_price = total_price(the_bike_id, the_q) #calculate total price
            bikeDetails = [the_q,ship,shipCost.replace('$','')] #store user details
            bike.update({the_bike_id:bikeDetails}) #update user details
            invoice(the_bike_id,bike, comp_name, comp_number, comp_email,
comp_location, current_date, ship, shipCost) #append bill
```

```
            add_bike_stock()

            comp_user() #print details


        #if input is no

        elif ask.upper() == "N":


print("\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")

            print(" Thank you!! Your selected bikes were added successfully.")


print("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n")

            run = False

            #loop terminates


         #input except yes and no

        else:

            print("\n+++++++++++++++++++++++++++++++++++++++++++++")

            print("Invalid Input!!! Please enter Y or N ")

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

```
#the given below function runs when user press 1 for purchase
def purchaseBike():
    global the_bike_id, bike, the_q, the_price
    print("\n\n----------------------")
    print("|  Purchase  Bikes   |")
    print("----------------------\n")
    the_bike_id = 0
    the_q = 0
    bike = {}
    buyMore = True
    while buyMore:
        user()
        the_bike_id = validating_bike_id() #validating bike id
        print("\n")
        the_q = quantity_validation(the_bike_id) #validating quantity
        bike.update({the_bike_id:the_q}) #updating user input


        final_sell(the_bike_id, the_q) #finalizing sell
        the_price = total_price(the_bike_id, the_q) #total price calculate
```

```
        purchase_bike() #purchase bike display
        bill(the_bike_id, bike, name, num, email, location, curr_date) #generating bill


        puser() #user details
        user_ask() #asking if user wants to add more
        print("\n")
        user_operation() #asking user to operate
        buyMore = False
        #loop terminates



#the given below function runs when user press 2 for add
def addStock():
    global the_bike_id, bike, the_q, the_price
    print("\n\n------------------")
    print("|   Add  Stock   |")
    print("------------------\n")
    the_bike_id = 0
    the_q = 0
    bike = {}
    bikeDetails = []
```

```
    addMore = True
    #loop runs
    while addMore:
        com_user()
        the_bike_id = validating_bike_id() #validating bike id
        the_q = quantity_val(the_bike_id) #validating quantity
        final_add(the_bike_id, the_q) #finalizing add
        the_price = total_price(the_bike_id, the_q) #total price calculate
        bikeDetails = [the_q,ship,shipCost.replace('$','')] #updating required parameters on
list
        add_bike_stock() #adding bike to stock
        bike.update({the_bike_id:bikeDetails}) #updating user input
        invoice(the_bike_id,bike, comp_name, comp_number, comp_email,
comp_location, current_date, ship, shipCost) #generating bill
        comp_user() #user details
        comp_ask() #asking if user wants to add more
        print("\n")
        user_operation() #asking user to operate
        addMore = False
        #loop terminates
```

```
#the given below function generates bill when user purchases bike
def bill(the_bike_id,bike, name, num, email, location, curr_date):
    global bike_id, b_curr
    #declaring global to access it from other function
    results = add_bike_2D_list()
    bike_id = the_bike_id
    b_name = name
    b_num = num
    b_email = email
    b_location = location
    b_curr = curr_date
    bbike = bike
            #function to extract current date and time
    dt = str(datetime.datetime.now().year) + "-" + str(datetime.datetime.now().month) + "-"
+ str(
        datetime.datetime.now().day) + "-" + str(datetime.datetime.now().hour) + "-" + str(
        datetime.datetime.now().minute) + "-" + str(datetime.datetime.now().second)
    invoice = str(dt)  # unique invoice
    t = str(datetime.datetime.now().year) + "-" + str(datetime.datetime.now().month) + "-"
+ str(
```

```
        datetime.datetime.now().day)  # date

    d = str(t)  # date

    u = str(datetime.datetime.now().hour) + ":" + str(datetime.datetime.now().minute) + ":"
+ str(

        datetime.datetime.now().second)  # time

    e = str(u)  # time

    grandTotal = 0

    with open("invoice-"+""+name+""+""+num+".txt", 'w') as f:

    #function to write in the text file

        f.write("                    Bike Management System                    \n")


f.write("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
\n")

        f.write("\n\nInvoice: " + invoice + "\t\tDate: " + d +"\n" +'\n\t\t\t\tTime: ' + e + "")

        f.write('\n'+"\nName of Customer: " + str(b_name) +"" + "\n")

        f.write("Phone Number: " + str(b_num)+"" + "\n")

        f.write("Email-address: " + str(b_email) +"" + "\n")

        f.write("Current Location: " + str(b_location) +"" + "\n" + '\n\nBikes Purchased:\n-----
--------------------------------------------------------\n')

    for i in bbike:

        with open("invoice-"+""+name+""+""+num+".txt", 'a') as f:

        #function to append the text file
```

```
            f.write('Bike Name: ' + results[i-1][0] + '\nCompany Name: ' + results[i-1][1] +
'\nBike Color: ' + results[i-1][2] + '\nQuantity: ' + str(bbike[i]) + '\nPrice of Single Bike: ' +
results[i-1][4] + '\nTotal Amount: $' + str(int(results[i-1][4].replace('$',''))*bike[i]) + '\n-------
--------------------------------------------------\n')


        grandTotal = grandTotal + int(results[i-1][4].replace('$',''))*bbike[i]

         #function to calculate grand total


    with open("invoice-"+""+name+""+""+num+".txt", 'a') as f:

    #function to append the text file with grand total

        f.write('\nGrand Total: $' + str(grandTotal))




#the given below function generates bill when user adds bike

def invoice(the_bike_id,bike, comp_name, comp_number, comp_email, comp_location,
current_date, ship, shipCost):

    global bike_id, dd

    results = add_bike_2D_list()

    bike_id = the_bike_id

    nam = comp_name

    numb = comp_number
```

```
    em = comp_email

    loc = comp_location

    dd = current_date

    sship = ship

    sshipCost = shipCost

    bbike = bike



            #function to extract current date and time

    dt = str(datetime.datetime.now().year) + "-" + str(datetime.datetime.now().month) + "-"
+ str(

        datetime.datetime.now().day) + "-" + str(datetime.datetime.now().hour) + "-" + str(

        datetime.datetime.now().minute) + "-" + str(datetime.datetime.now().second)

    invoice = str(dt)  # unique invoice

    t = str(datetime.datetime.now().year) + "-" + str(datetime.datetime.now().month) + "-"
+ str(

        datetime.datetime.now().day)  # date

    d = str(t)  # date

    u = str(datetime.datetime.now().hour) + ":" + str(datetime.datetime.now().minute) + ":"
+ str(

        datetime.datetime.now().second)  # time

    e = str(u)  # time

    grandTotal = 0
```

```
    with open("invoice-"+""+comp_name+""+""+comp_number+".txt", 'w') as f:

    #function to write in the text file


        f.write("                    Bike Management System                    \n")


f.write("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
\n")

        f.write("\n\nInvoice: " + invoice + "\t\tDate: " + d +"\n" +'\n\t\t\t\tTime: ' + e + "")

        f.write('\n'+"\nName of Customer: " + str(nam) +"" + "\n")

        f.write("Phone Number: " + str(numb)+"" + "\n")

        f.write("Email-address: " + str(em) +"" + "\n")

        f.write("Current Location: " + str(loc) +"" + "\n" + '\n\nBikes Added:\n--------------------
-------------------------------------\n')


    for i in bbike:

        with open("invoice-"+""+comp_name+""+""+comp_number+".txt", 'a') as f:

        #function to append the text file


            f.write('Bike Name: ' + results[i-1][0] + '\nCompany Name: ' + results[i-1][1] +
'\nBike Color: ' + results[i-1][2] + '\nQuantity: ' + str(bbike[i][0]) + '\nPrice of Single Bike: '
+ results[i-1][4] + '\nTotal Amount: $' + str(int(results[i-1][4].replace('$',''))*bike[i][0]) +
'\nShipping Company: ' + bbike[i][1] + '\nShipping Cost: $' + bbike[i][2] +'\n-------------------
-------------------------------------\n')
```

```
        grandTotal = grandTotal + int(results[i-1][4].replace('$',''))*bbike[i][0] +
int(bbike[i][2])

        #function to calculate grand total


    with open("invoice-"+""+comp_name+""+""+comp_number+".txt", 'a')  as f:

    #function to append the text file with grand total

        f.write('\nGrand Total: $' + str(grandTotal))
```

APPENDIX B: MAIN FILE

```
#importing function file

import function


function.welcome()

#calling to print welcome message from function file


function.display_bike()

#calling to display bike from function file


function.user_operation()

#calling user operation from function file


#the given below function checks if the input is 1,2 or 3 or any other numbers

def check_user_input():

    loop = True

    #run while loop

    while loop == True:

        try:

            user_input = int(input("Enter the number: "))
```

```
        #checking user input
        if user_input == 1:
            function.purchaseBike()
            #calling purchase bike function


        elif user_input == 2:
            function.addStock()
            #calling add bike function


        elif user_input == 3:
            function.exit_system()
            #calling exit system function
            loop = False
            #loop terminates
        else:
            function.invalid_user_input()
            #calling to display invalid input function
    except ValueError:
        #value error if any other than integer is entered
        print("\n+++++++++++++++++++++++++++++++++++++++++++")
        print("Please provide Integer Value!!!!")
```

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


check_user_input()


#calling function to run
```