



**Islington college**  
(इस्लिंग्टन कलेज)

**Module Code & Module Title**

**CC4059NI Fundamentals of Computing**

**Assessment Weightage & Type**

**60% Individual Coursework**

**Year and Semester**

**2021-22 Autumn**

**Student Name: Manjil Koju**

**Group: MAD**

**London Met ID: 21049365**

**College ID: np01ma4a210003**

**Assignment Due Date: 26<sup>th</sup> August/2022**

**Assignment Submission Date: 26<sup>th</sup> August/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

1. Introduction .....	1
1.1. About the project .....	1
1.2. Goals and Objective .....	1
2. Discussion and Analysis .....	2
2.1. Algorithm .....	2
2.2. Flowchart .....	5
2.3. Pseudocode .....	8
2.4. Data Structure .....	24
3. Program .....	27
3.1. Option1: Renting process .....	27
3.2. Option2: Returning process .....	32
3.3. Option3: Exit .....	36
4. Testing .....	37
4.1. Test 1 .....	37
4.2. Test 2 .....	39
4.3. Test 3 .....	42
4.4. Test 4 .....	48
4.5. Test 5 .....	52
5. Conclusion .....	60
References .....	61
Appendix .....	62
• Code for main.py .....	62
• Code for operations.py .....	63
• Code for return_operations.py .....	72

- Details in costume\_details.txt ..... 77

## Table of Tables:

Table 1: Test1; implementation of try, except.....	37
Table 2: Test 2; providing negative or non-existing values.....	39
Table 3 Test 3; complete renting process: .....	42
Table 4: Test4; complete returning process .....	48
Table 5: Test5; showing change in quantity after renting and returning .....	52

## Table of Figures:

Figure 1: Flowchart 1.....	5
Figure 2: Flowchart 2.....	6
Figure 3: Flowchart 3.....	7
Figure 4: Use of dictionary to store costume details.....	25
Figure 5: Creating dictionary to store records from the rent bill.....	26
Figure 6: Creating a 2D list to store the list of costumes being rented .....	26
Figure 7: Appending values into the list.....	26
Figure 8: Start-up screen.....	27
Figure 9: entering option1 for renting .....	28
Figure 10: Entering the name, id and quantity.....	28
Figure 11: Asking user to rent more and entering y to rent more .....	29
Figure 12: Entering id and quantity and entering n to complete the process .....	29
Figure 13: Displaying the rent bill .....	30
Figure 14: Rent bill in text file .....	31
Figure 15: Enter option2 for returning.....	32
Figure 16: Entering name, bill no and rent days .....	33
Figure 17: Displaying the return bill with fine.....	34
Figure 18: Return bill as text file with fine.....	35
Figure 19: Entering option3 and the program terminates .....	36
Figure 20: Entering the option and name .....	37
Figure 21: Entering string as id .....	38
Figure 22: Error message was shown .....	38
Figure 23: Entering the option and name .....	40
Figure 24: Entering negative value.....	41
Figure 25: Entering non-existing value .....	41
Figure 26: Entering option and name for renting .....	43
Figure 27: Entering id and quantity .....	43
Figure 28: Choosing to rent more.....	44
Figure 29: Entering the id and quantity second time .....	44
Figure 30: Completing the renting process.....	45

Figure 31: Rent bill being displayed .....	46
Figure 32: Rent bill in text file .....	47
Figure 33: Choosing option to return .....	48
Figure 34: Entering name and bill no .....	49
Figure 35: Entering no. of days .....	49
Figure 36: Displaying the bill for return with fine.....	50
Figure 37: Bill for return in text file.....	51
Figure 38: Entering option for renting and name .....	53
Figure 39: Entering the id and quantity.....	53
Figure 40: Choosing to rent more.....	54
Figure 41: Entering id and quantity again.....	54
Figure 42: Original costume details .....	55
Figure 43: Costume details after renting .....	55
Figure 44: Original costume details in text file.....	56
Figure 45: Costume details in text file after renting .....	56
Figure 46: Entering the option for returning.....	57
Figure 47: Entering name, bill no and no. of days .....	57
Figure 48: Costume details before returning .....	58
Figure 49: Costume details after returning, back to original .....	58
Figure 50: Costume details before returning in text file .....	59
Figure 51: Costume details after returning in text file .....	59

## 1. Introduction

### 1.1. About the project

The coursework was about making an application for a Costume Rental Shop that stores information about the available costumes in a text file. The purpose of the application is to read the details from the text file and allows the user to rent the costumes based on their availability. Also, the given text file containing the costume details should be updated after each transaction of renting or returning. A bill is also to be generated for each renting of costumes and returning as well. The rental shop requires an application that can generate unique bill for each transaction and also implement a fine system when the costumes are not returned within a certain time limit. Along with generating the bill for the transactions, the costume details should also be updated like decreasing of increasing the quantity of costumes after renting or returning.

### 1.2. Goals and Objective

The main objective of this project is to create an application for the Costume Rental Shop. The application after creation should be able to perform the following tasks:

- To allow the user to rent the costumes based on their availability.
- To allow the user to return the rented costumes and also implement a fine if not returned within the time limit.
- To provide user with smoother experience by handling the invalid input from the user through appropriate error messages.
- To be able to generate and display bills for each renting and returning process.

## 2. Discussion and Analysis

### 2.1. Algorithm

STEP 1: Display welcome message.

STEP 2: Display the options to the user.

STEP 3: Ask user to input an option and assign it to user\_input

STEP 4: If user\_input is 1, go to STEP 5; if user\_input is 2, go to STEP 21; if user\_input is 3, go to STEP 38; otherwise go to STEP 2.

STEP 5: Display the available costumes from the text file containing costume details.

STEP 6: Ask user to enter their name and assign it to name.

STEP 7: If name is 'back', go to STEP 2; otherwise go to STEP 8.

STEP 8: Ask user to enter the id of the costume to rent and assign it to input\_id.

STEP 9: If input\_id is invalid or not available, go to STEP 8; otherwise go to STEP 10.

STEP 10: Ask user to enter the quantity of costume to rent and assign it to quantity.

STEP 11: If quantity is not available or invalid, go to STEP 10; otherwise assign the costume details from the text file to a dictionary of name costume\_dictionary and update the dictionary by subtracting quantity from the available quantity.

STEP 12: Re-write the updated costume\_dictionary into the text file containing costume details.

STEP 13: Extract the price of the costume from costume\_dictionary and assign it to price.

STEP 14: Multiply price with quantity and assign the value to total\_price then display the price to the user

STEP 15: Create a list of name item\_list with empty value.

STEP 16: Add the costume details which includes costume name, brand name, price and quantity into item\_list as a row.

STEP 17: Display the available costumes from the text file containing costume details and ask the user if they want to rent more costumes. If the answer is yes, go to STEP 8; otherwise go to STEP 18.

STEP 18: Display the bill that contains the name, date and time, total\_price and the list of costumes that has been rented, this list is extracted from item\_list.

STEP 19: Also display the detail about fine system and the bill no that has been generated to the user.



STEP 20: Create a text file which contains the details the same as the bill that had been displayed in STEP 18.

STEP 21: The process of renting is complete so go to STEP 3.

STEP 22: Display the available costumes to user by extracting from the text file containing costume details.

STEP 23: Ask user to enter their name and assign it to name.

STEP 24: If name is 'back', go to STEP 2; otherwise go to STEP 25.

STEP 25: Ask user to enter their bill number and assign it to bill\_no.

STEP 26: Check if a text bill for the input values of name and bill\_no exists. If it exists, go to STEP 27; otherwise go to STEP 23.

STEP 27: From the existing text bill, extract the details like costume name, brand name, quantity and price, and then assign the values to a dictionary of name record.

STEP 28: Also extract the total price from the text bill and assign it to total\_price.

STEP 29: Ask user to enter the number of renting days and assign the value to rent\_days.

STEP 30: If the input value for rent\_days is not valid or negative, go to STEP 29; otherwise goto STEP 31.

STEP 31: Extract the costume details form text file and store it in a dictionary of name costume\_dictionary.

STEP 32: Compare the costume name from costume\_details and record and if they are same, increase the quantity in costume\_details by adding the quantity from record.

STEP 33: Re-write the updated costume\_dictionary into the text file containing costume details.

STEP 34: If rent\_days is greater than 5, subtract 5 from rent\_days and multiply it with 7.5 then assign the value to fine; otherwise assign value 0 to fine.

STEP 35: Display the return bill that contains name, date and time, rent\_days, the costume details from record and total price but if fine is not equal to 0 also display the fine and sum of fine and total price.

STEP 36: Create a text file which contains the details the same as the return bill that had been displayed in STEP 35.

STEP 37: The process of returning is complete so go to STEP 3.

STEP 38: Display a 'Thank You' message to the user.

STEP 39: End the program.

## 2.2. Flowchart

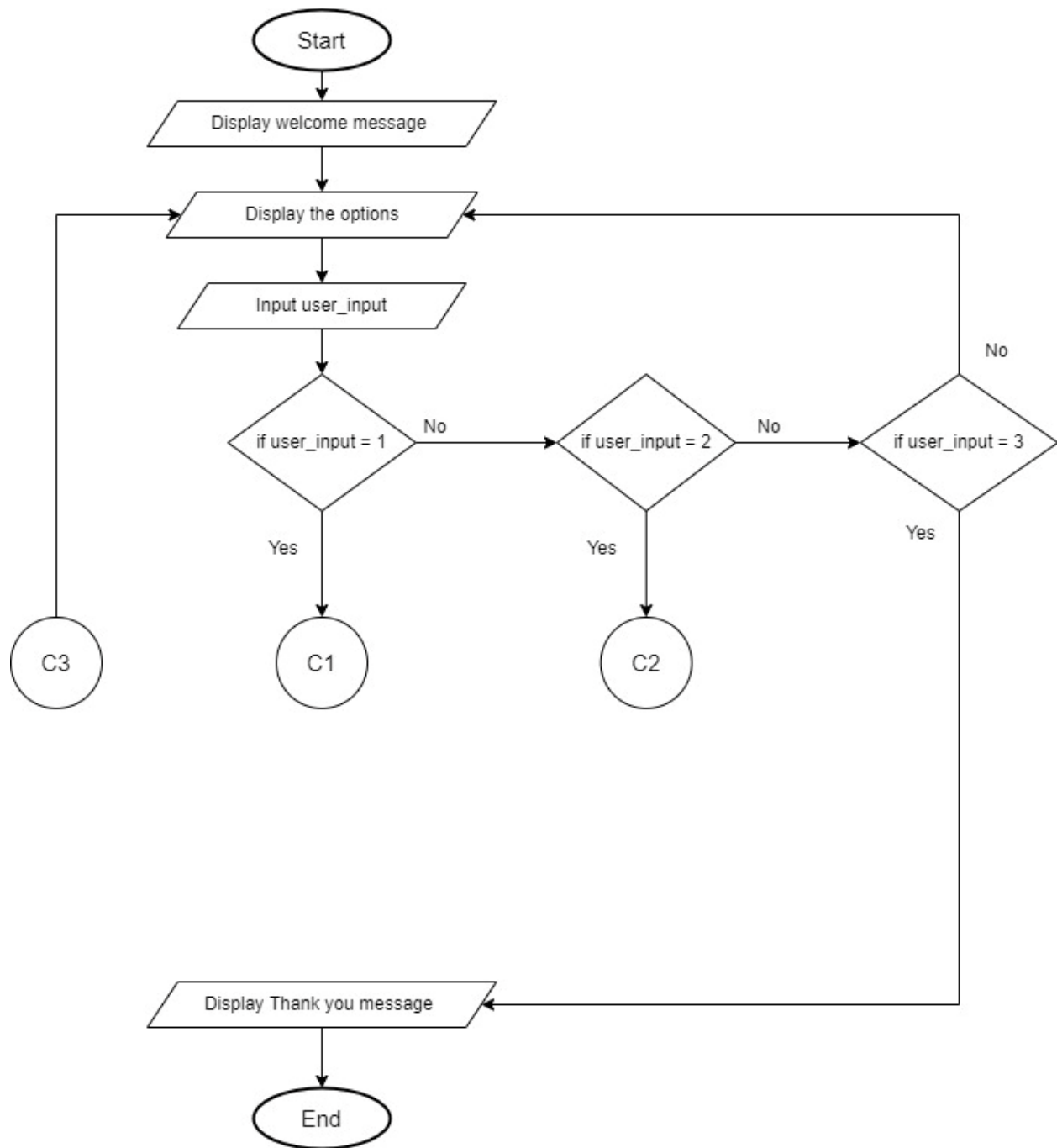


Figure 1: Flowchart 1

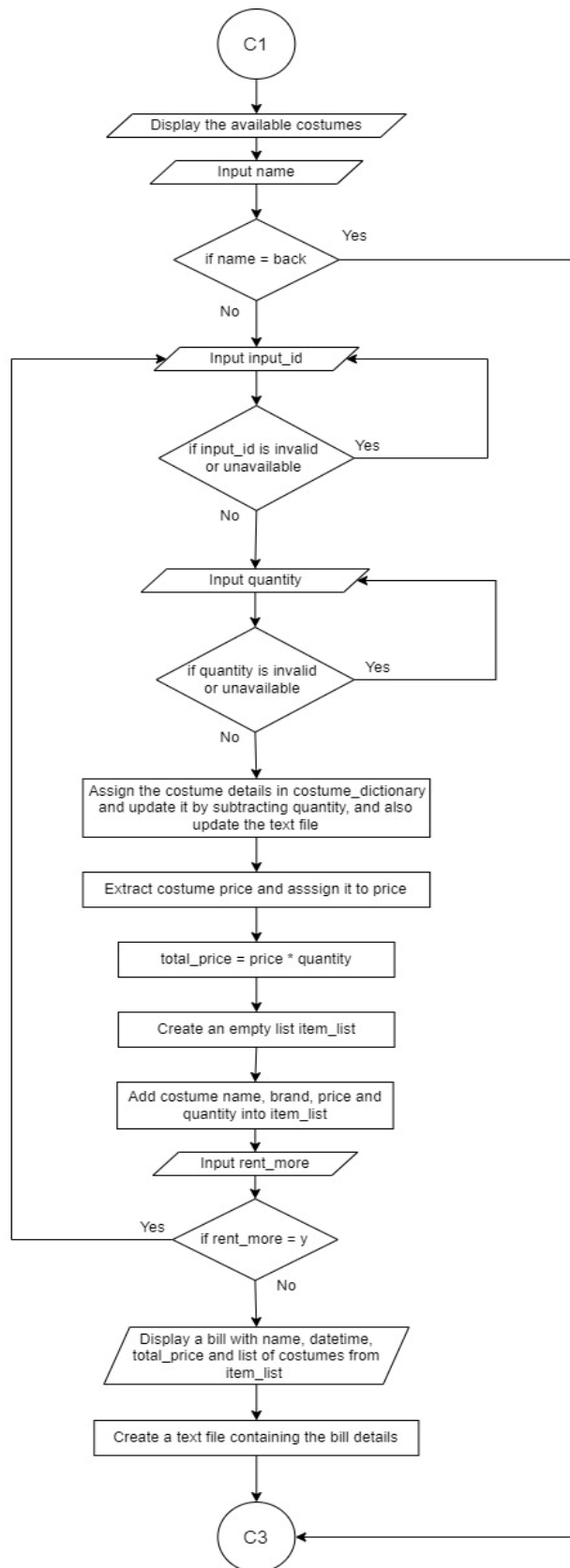


Figure 2: Flowchart 2

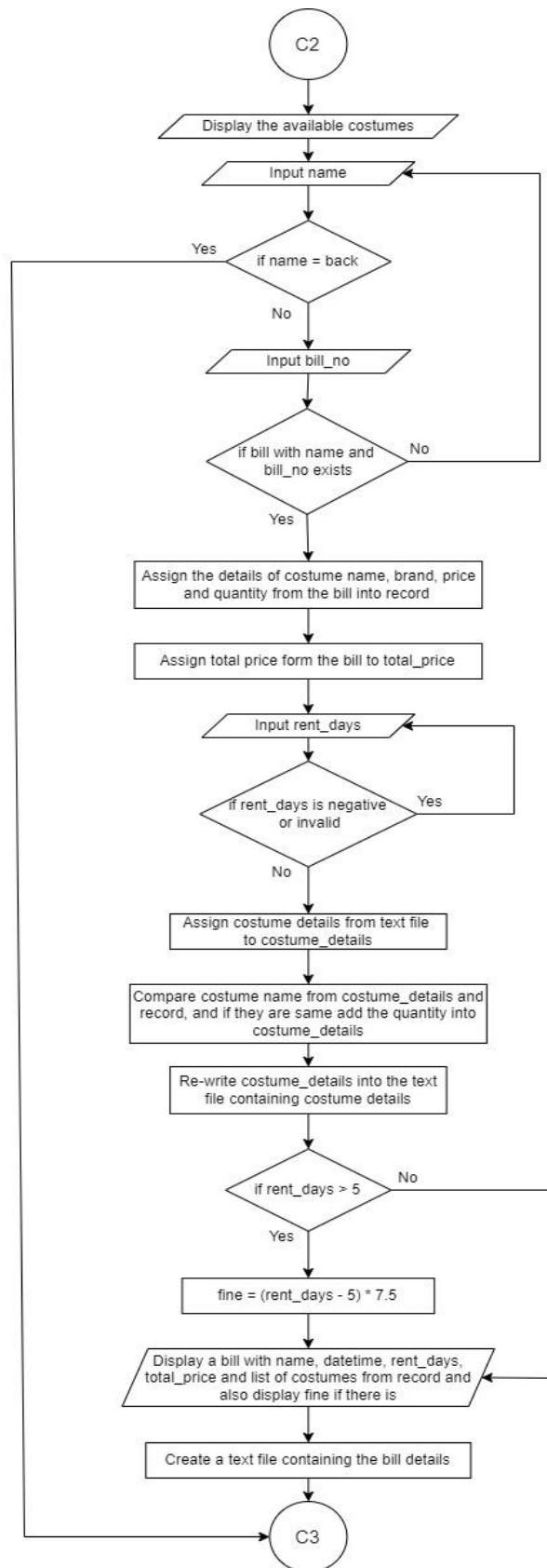


Figure 3: Flowchart 3

### 2.3. Pseudocode

- Pseudocode for main.py

```
FROM operations IMPORT options, option_1, option_2, option_3, invalid_option
OUTPUT ("-----")
OUTPUT ("      Welcome to Costume Rental Application")
OUTPUT ("-----")
OUTPUT ("\n")
SET run TO True
WHILE run
    CALL options()
    SET user_input TO INPUT("Enter your option: ")
    OUTPUT ("\n")
    IF user_input == "1"
        CALL option_1()
    ELSE IF user_input == "2"
        CALL option_2()
    ELSE IF user_input == "3"
        CALL option_3()
        SET run TO False
    ELSE
        CALL invalid_option()
    END IF
END WHILE
```

- Pseudocode for operations.py

**IMPORT** datetime, random

**IMPORT** return\_operations

**DEFINE FUNCTION** costumes()

**OUTPUT** ("-----  
-")

**OUTPUT** ("ID      Costume Name              Company Name              Rent Price              Quantity")

**OUTPUT** ("-----  
-")

**OPEN** ("costume\_details.txt","r") **AS** file

**SET** id **TO** 1

**FOR** line **IN** file

**OUTPUT** (id, "\t" + line.replace(", ", "\t\t"))

**SET** id **TO** id +1

**END FOR**

**OUTPUT** ("\n")

**CLOSE** file

**END FUNCTION**

**DEFINE FUNCTION** dictionary\_creation()

**OPEN** ("costume\_details.txt","r") **AS** file

**SET** costume\_dictionary **TO** {}

**SET** costume\_id **TO** 1

**FOR**

**SET** line **TO** line.replace("\n", "")

        costume\_dictionary.update({costume\_id:(line.split(","))})

**SET** costume\_id **TO** costume\_id +1

**END FOR**

**CLOSE** file

**RETURN** costume\_dictionary

**END FUNCTION**

```
DEFINE FUNCTION text_update(costume_dictionary)
  OPEN ("costume_details.txt","w") AS file
  FOR key,value IN costume_dictionary.items()
    file.write(value[0] + "," + value[1] + "," + value[2] + "," + value[3])
    file.write("\n")
  END FOR
  CLOSE file
END FUNCTION
```

```
DEFINE FUNCTION quantity_update(input_id,quantity)
  SET costume_dictionary TO dictionary_creation()
  SET available_quantity TO int(costume_dictionary[input_id][3])
  WHILE quantity<=0 OR quantity>available_quantity
    IF quantity>available_quantity
      OUTPUT ("Quantity provided is greater than what we have in stock.")
    ELSE
      OUTPUT ("Enter a valid quantity.")
    END IF
    OUTPUT ("\n")
    SET success TO False
    WHILE success == False
      TRY
        SET quantity TO int(INPUT("Enter the quantity of costume: "))
        SET success TO True
      EXCEPT
        OUTPUT ("\n")
        OUTPUT ("+++++")
        OUTPUT ("Invalid value for quantity! Please enter a number.")
        OUTPUT ("+++++")
        OUTPUT ("\n")
      END WHILE
```



```

    OUTPUT ("n")
END WHILE
SET costume_dictionary[input_id][3] TO str(available_quantity - quantity)
CALL text_update(costume_dictionary)
RETURN costume_dictionary,quantity
END FUNCTION

DEFINE FUNCTION generate_bill(name,total_price,item_list,bill_no)
    SET SN TO 0
    OPEN ("Rent_"+name.lower()+"_"+bill_no+".txt","w") AS file
    file.write("Name of customer: "+name)
    file.write("n")
    file.write("Date Time of borrow: "+str(datetime.datetime.now()))
    file.write("n n")
    file.write("SN \t\t Costume Name \t\t Brand \t\t Unit Price \tQuantity")
    file.write("n")
    file.write("-----")
    file.write("n")
    FOR each IN item_list
        SET SN TO SN + 1
        file.write(str(SN)+"\t\t\t "+each[0)+"\t\t\t "+each[1)+"\t\t\t "+str(each[2])+"\t\t\t "+str(each[3]))
        file.write("n")
    END FOR
    file.write("-----")
    file.write("n n")
    file.write("Total price is: "+str(total_price))
    file.write("n")
    CLOSE file
END FUNCTION

```

```

DEFINE FUNCTION bill(name,total_price,item_list)
    SET SN TO 0
    SET bill_no TO str(random.randint(0,10000))
    OUTPUT ("=====")
    OUTPUT ("          Bill Details")
    OUTPUT ("=====")
    OUTPUT ("*****")
    OUTPUT ("\n")
    OUTPUT ("Name of customer:",name)
    OUTPUT ("Date Time of rent:",datetime.datetime.now())
    OUTPUT ("\n")
    OUTPUT ("SN \t Costume Name \t\t Brand \t\t Unit Price \t Quantity")
    OUTPUT ("-----")
    FOR each IN item_list
        SET SN TO SN + 1
        OUTPUT (str(SN)+"\t"+each[0]+\t\t+each[1]+\t\t " "+str(each[2])+"\t\t "+str(each[3]))
        OUTPUT ("\n")
    END FOR
    OUTPUT ("-----")
    OUTPUT ("\n")
    OUTPUT ("Total price:",total_price)
    OUTPUT ("\n")
    OUTPUT ("*****")
    OUTPUT ("\n")
    CALL generate_bill(name,total_price,item_list,bill_no)
    OUTPUT ("-----")
    OUTPUT ("|      Rent bill has been generated.      |")
    OUTPUT ("|      Your bill no is:",bill_no,"          |")
    OUTPUT ("-----")
    OUTPUT ("\n")
    OUTPUT ("** Note: If the costumes are not returned within 5 days,  **")

```

```

OUTPUT ("**      there will be additional fine of $7.5 per day      **")
OUTPUT ("\n")
END FUNCTION

```

```

DEFINE FUNCTION valid_id()
  SET success TO False
  WHILE success == False
    TRY
      SET input_id TO int(INPUT("Enter the ID of the costume: "))
      SET success TO True
    EXCEPT
      OUTPUT ("\n")
      OUTPUT ("+++++")
      OUTPUT ("Invalid value for ID! Please enter a number.")
      OUTPUT ("+++++")
      OUTPUT ("\n")
  END WHILE

  WHILE input_id <= 0 OR input_id > len(dictionary_creation)
    OUTPUT ("\n The input Costume ID in invalid! Please enter a valid ID. \n")
    SET success TO False
    WHILE success == False
      TRY
        SET input_id TO int(INPUT("Enter the ID of the costume: "))
        SET success TO True
      EXCEPT
        OUTPUT ("\n")
        OUTPUT ("+++++")
        OUTPUT ("Invalid value for ID! Please enter a number.")
        OUTPUT ("+++++")
        OUTPUT ("\n")
    END WHILE

```

```

END WHILE
OUTPUT ("\n")
OUTPUT ("+++++")
OUTPUT ("Costume is available")
OUTPUT ("+++++")
OUTPUT ("\n")
RETURN input_id
END FUNCTION

```

```

DEFINE FUNCTION renting(name)
  SET rent_more TO True
  SET item_list TO []
  SET SN TO 0
  SET total_price TO 0
  WHILE rent_more
    SET input_id TO valid_id
    SET success TO False
    WHILE success == False:
      TRY
        SET quantity TO int(INPUT("Enter the quantity of costume: "))
        SET success TO True
      EXCEPT
        OUTPUT ("\n")
        OUTPUT ("+++++")
        OUTPUT ("Invalid value for quantity! Please enter a number.")
        OUTPUT ("+++++")
        OUTPUT ("\n")
    END WHILE
    OUTPUT ("\n")
    SET updating_dictionary TO quantity_update(input_id,quantity)
    SET costume_dictionary TO updating_dictionary[0]

```

```

SET quantity TO updating_dictionary[1]
SET price TO float((costume_dictionary[input_id][2]).replace("$",""))
SET total_price TO total_price + price * quantity
OUTPUT ("The price is:",price)
OUTPUT ("\n")
item_list.append([costume_dictionary[input_id][0],costume_dictionary[input_id][1],
price,quantity])
CALL costumes()
OUTPUT ("Rent another costume as well?")
SET more TO INPUT ("Please enter 'y' to rent more costume or any other key to
stop.")
OUTPUT ("\n")
IF more.lower() != "y"
    SET rent_more TO False
END IF
END WHILE
CALL bill(name,total_price,item_list)
END FUNCTION

DEFINE FUNCTION options()
    OUTPUT ("Please select an option:")
    OUTPUT ("(1) || Press 1 to rent costumes.")
    OUTPUT ("(2) || Press 2 to return costumes.")
    OUTPUT ("(3) || Press 3 to exit.")
    OUTPUT ("\n")
END FUNCTION

DEFINE FUNCTION option_1()
    OUTPUT ("+++++
+++++")
    OUTPUT (" Let's rent a costume. You can enter 'back' as name to go back.")

```

```

OUTPUT ("+++++
+++++")
OUTPUT ("
")
CALL costumes()
OUTPUT ("
")
SET name TO INPUT("Enter your name: ")
OUTPUT ("
")
IF name.lower() != "back"
    CALL renting(name)
END IF
END FUNCTION

```

```

DEFINE FUNCTION option_2()
    OUTPUT ("+++++
+++++")
    OUTPUT (" Let's return a costume. You can enter 'back' as name to go back.")
    OUTPUT ("+++++
+++++")
    OUTPUT ("
")
    CALL costumes()
    CALL return_operations.returning()
END FUNCTION

```

```

DEFINE FUNCTION option_3()
    OUTPUT ("      Thank you for using the application.")
    OUTPUT ("
")
END FUNCTION

```

```

DEFINE FUNCTION invalid_option()
    OUTPUT ("Invalid Option!!!!")
    OUTPUT ("Please select the value as per the given options.")

```

```
    OUTPUT ("\n")  
END FUNCTION
```

- Pseudocode for return\_operations.py

```
IMPORT datetime
IMPORT operations
DEFINE FUNCTION ask_rent_days()
    SET days_success TO False
    WHILE days_success == False
        TRY
            SET positive TO False
            WHILE positive == False
                SET rent_days TO int(INPUT("Enter the no of renting days: "))
                IF rent_days > 0
                    SET positive TO True
                ELSE
                    OUTPUT ("\n")
                    OUTPUT ("Invalid value for no of renting days! Please enter a positive number.")
                    OUTPUT ("\n")
                END IF
            END WHILE
            SET days_success TO True
        EXCEPT
            OUTPUT ("\n")
            OUTPUT ("Invalid value for no of renting days! Please enter a number.")
            OUTPUT ("\n")
    END WHILE
    RETURN rent_days
END FUNCTION

DEFINE FUNCTION return_bill_text(name,bill_no,record,total_price,rent_days,fine)
    OPEN ("Return_"+name.lower()+"_"+bill_no+".txt","w") AS file
    file.write("Name of customer: "+name)
```



```

file.write("\n")
file.write("Date Time of return: "+str(datetime.datetime.now()))
file.write("\n")
file.write("No of renting days: "+str(rent_days))
file.write("\n\n")
file.write("SN \t\t Costume Name \t\t Brand \t\t Unit Price \tQuantity")
file.write("\n")
file.write("-----")
file.write("\n")
FOR each IN record.values
    file.write(str(each[0])+"\t\t " +each[1]+"\t\t " +each[2]+"\t\t " +str(each[3])+"\t\t "
    +str(each[4]))
    file.write("\n")
END FOR
file.write("-----")
file.write("\n\n")
file.write("Total price is: "+str(total_price))
file.write("\n")
IF fine !=0
    file.write("\n")
    file.write("** Since the costumes were returned "+str(rent_days-5)+" days late;")
    file.write("\n")
    file.write("Fine: "+str(fine))
    file.write("\n")
    file.write("Total price with fine: "+str(fine+total_price))
    file.write("\n")
END IF
CLOSE file
END FUNCTION

```

**DEFINE FUNCTION** return\_bill(name,bill\_no,record,total\_price,rent\_days)

```

SET fine TO 0
OUTPUT ("=====")
OUTPUT ("          Bill Details")
OUTPUT ("=====")
OUTPUT ("*****")
OUTPUT ("\n")
OUTPUT ("Name of customer:",name)
OUTPUT ("Date Time of return:",datetime.datetime.now())
OUTPUT ("No of renting days:",rent_days)
OUTPUT ("\n")
OUTPUT ("SN \t Costume Name \t\t Brand \t\t Unit Price \t Quantity")
OUTPUT ("-----")
FOR each in record.values()
    OUTPUT (str(each[0]) + "\t"+each[1] + "\t\t"+each[2]+\t    " +str(each[3]) + "\t\t ",
    str(each[4]))
    OUTPUT ("\n")
END FOR
OUTPUT ("-----")
OUTPUT ("\n")
OUTPUT ("Total price:",total_price)
IF rent_days > 5
    SET fine TO 7.5 * (rent_days - 5)
    OUTPUT ("\n")
    OUTPUT ("** Since the costumes were returned",rent_days-5,"days late;")
    OUTPUT ("Fine:",fine)
    OUTPUT ("Total price with fine:",total_price + fine)
END IF
OUTPUT ("\n")
OUTPUT ("*****")
OUTPUT ("\n")
CALL return_bill_text(name,bill_no,record,total_price,rent_days,fine)

```

```

OUTPUT ("-----")
OUTPUT ("|      Return bill has been generated.      |")
OUTPUT ("-----")
OUTPUT ("\n")

```

**END FUNCTION**

```

DEFINE FUNCTION return_update(record)
  SET costume_dictionary TO operations.dictionary_creation
  FOR value in costume_dictionary.values()
    FOR records in record.values()
      IF value[0] == records[1]
        SET value[3] TO str(int(value[3])+int(records[4]))
      END IF
    END FOR
  END FOR
  CALL operations.text_update(costume_dictionary)
END FUNCTION

```

```

DEFINE FUNCTION details_from_bill(name,bill_no)
  OPEN ("Rent_"+name.lower()+"_"+bill_no+".txt","r") AS file
  SET iteration TO 0
  SET lines TO list(file)
  SET record TO {}
  SET rID TO 0
  SET total_price TO 0
  FOR line in lines
    SET iteration TO iteration + 1
    IF iteration>5 and iteration<(len(lines)-2)
      SET rID TO rID + 1
      SET line TO line.replace("\n","")
      record.update({rID:line.split("\t\t")})
    END IF
  END FOR

```

```
    END IF
    IF iteration == len(lines)
        SET total_price TO float(line.replace("Total price is: ",""))
    END IF
END FOR
CLOSE file
RETURN record,total_price
END FUNCTION

DEFINE FUNCTION returning()
    SET success TO False
    WHILE success == False
        SET name TO INPUT("Enter your name: ")
        IF name.lower() == "back"
            BREAK
        END IF
        SET bill_no TO INPUT("Enter your bill no: ")
        OUTPUT ("\n")
        TRY
            OPEN ("Rent_"+name.lower()+"_"+bill_no+".txt","r") AS file
            CLOSE file
            SET success TO True
        EXCEPT
            OUTPUT ("The input customer name or bill no is incorrect! Please enter the
            correct value.")
            OUTPUT ("\n")
        END WHILE
    IF name.lower() != "back":
        SET bill_details TO details_from_bill(name,bill_no)
        SET record TO bill_details[0]
        SET total_price TO bill_details[1]
```

```
    SET rent_days TO ask_rent_days()  
    CALL return_update(record)  
    CALL return_bill(name,bill_no,record,total_price,rent_days)  
END IF  
END FUNCTION
```

## 2.4. Data Structure

Data structures are containers that are used for storing, organizing and retrieving data. Data structures make it easy for the users to access and manipulate the data in an appropriate way and are used for efficient data persistence (Loshin, 2021). The data structures in python are either mutable or non-mutable; mutable means that they can be modified or changed whereas immutable means they cannot be modified or changed after they are created. The basic python data structures include the following:

- List

List is an ordered collection of data and the data stored in a list does not have to be of the same datatype. Since a list is an ordered collection of data, each of the items in the list has a unique index which can be used to access them. Lists are mutable, meaning the items in the list can be changed, added, replaced or deleted even after the creation of the list. A list can also be nested i.e. the user can add a whole list as an item in another list, a 2D list is created this way. In python, a list is denoted by square brackets [].

- Set

Set is a collection of unique data that does not have a specific order. The elements in a set don't have any position or index. Sets are also mutable so the elements in a set can be added, deleted, changed or replaced. A set stores only unique data so the elements in a set cannot be duplicated. Because of this, sets are generally used to prevent the duplication of data. In python, a set is denoted by curly brackets {}.

- Tuples

Tuples are ordered collection of elements. It can store items of multiple datatype. It is very similar to list except tuples are immutable i.e. the data stored in a tuple cannot be changed or modified after the tuple is created. Just like a list, the elements in a tuple have a unique index which can be used to access them. In python, a tuple is denoted by round brackets ().

- Dictionary

A dictionary is an unordered collection of data but unlike other data structures, which store single value as single element, a dictionary stores a pair of key and value as a single element. In a dictionary, the keys are always unique but the values can be repeated and can also be of different datatypes. A dictionary is mutable, so the elements can be added, deleted, changed and replaced. The elements of a dictionary can be accessed using the keys. In python, a dictionary is denoted by curly brackets {} which is the same as set but a set does not contain key whereas a dictionary does.

➤ Data structures used in the code:

Among the above mentioned data structures, the ones used in this coursework are dictionary and list. A dictionary was used in the code to store the details of the costumes extracted from the text file. Each of the values in the dictionary were stored as a list that contained the costume name, brand, price and quantity. A dictionary was used in this case because it allows you to assign a unique key for each values. This makes it easier differentiate each of the costumes and modify the values when renting and returning operations are carried out.

```
def dictionary_creation():  
    """stores the costume details in a dictionary and returns it"""  
    file = open("costume_details.txt", "r")  
    costume_dictionary = {}  
    costume_id = 1  
    #reading from the text file and storing the details in a dictionary  
    for line in file:  
        line = line.replace("\n", "")  
        costume_dictionary.update({costume_id: (line.split(", "))})  
        costume_id += 1  
    file.close()  
    return costume_dictionary
```

Figure 4: Use of dictionary to store costume details

Similarly, a dictionary was also used to store the records from the rent bill in order to carry out the returning process. This was used in order to compare the costume list from the

bill and the previous mentioned dictionary for costume details and modify them accordingly after the process of returning.

```
def details_from_bill(name,bill_no):
    file = open("Rent_"+name.lower()+"_"+bill_no+".txt","r")
    iteration = 0
    lines = list(file)
    record = {}
    rID = 0
    total_price = 0
    for line in lines:#extracting the records from the rent bill and storing in
        iteration += 1
        if iteration>5 and iteration<(len(lines)-2):
            rID += 1
            line = line.replace("\n","")
            record.update({rID:line.split("\t\t ")})
        if iteration == len(lines):
            total_price = float(line.replace("Total price is: ",""))
    file.close()
    return record,total_price
```

Figure 5: Creating dictionary to store records from the rent bill

Lastly, a 2D list was also used in the code to store the list of costumes that are the user has rented. In the 2D list each row contains the costume name, brand, price and quantity of the costumes that are being rented so that they can be easily accessed while creating a rent bill.

```
def renting(name):
    """for renting costumes"""
    rent_more = True
    item_list = []
    SN = 0
    total price = 0
```

Figure 6: Creating a 2D list to store the list of costumes being rented

```
costume_dictionary = updating_dictionary[0]
quantity = updating_dictionary[1]
price = float((costume_dictionary[input_id][2]).replace("$",""))
total_price += price*quantity
print("The price is:",price)
print("\n")
item_list.append([costume_dictionary[input_id][0],costume_dictionary[input_id][1],price,quantity])
costumes()
print("Rent another costume as well?")
```

Figure 7: Appending values into the list



### 3. Program

At the start of execution of the program, the user should be displayed a welcome message and the three available options. Option1 is the option for renting costumes, Option2 is for returning costumes and Option3 is for exiting the program. If the user input any other value a message is displayed prompting them to choose the available ones.

```
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>
= RESTART: C:\Users\user\assign\Fundamental of computing\courseWork\weekly\main.py

-----
Welcome to Costume Rental Application
-----

Please select an option:
(1) || Press 1 to rent costumes.
(2) || Press 2 to return costumes.
(3) || Press 3 to exit.

Enter your option:
```

Figure 8: Start-up screen

#### 3.1. Option1: Renting process

For the renting process, the user should enter Option1. When this option is entered, the user is displayed the available costumes and then asked to enter their name. The user can also enter 'back' as the name to go to the previous step that displays the options. After the user enters their name they are asked to enter the id and quantity of the costumes they want to rent. If an invalid or non-existing input is given, a simple invalid message is displayed without the termination of the program. Then, the user is asked if they want to rent more. Here, the user can choose to either rent more costumes, in which case they are asked about id and quantity again, or to complete the renting process. The user can rent as many costumes as they want and as many times as they want just as long as they are available. After the renting process is complete, a bill is displayed to the user along with the fine system and the bill no. At the same time, a text file for the bill is also generated.

## Screenshots for renting process:

```
Enter your option: 1

+++++
Let's rent a costume. You can enter 'back' as name to go back.
+++++

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume       Cartmax Inc       $15.5           49
2       Formal Suite         Megaplex          $14.5           40
3       Fairy Costume        DollarSmart       $18             35
4       Thor Costume         Marvel Inc        $23             55
5       Princess dress       Disney LTD        $41             25
-----
```

Figure 9: entering option1 for renting

```
Enter your name: mike

Enter the ID of the costume: 1

+++++
Costume is available
+++++

Enter the quantity of costume: 4
```

Figure 10: Entering the name, id and quantity

ID	Costume Name	Company Name	Rent Price	Quantity
1	Cop Costume	Cartmax Inc	\$15.5	45
2	Formal Suite	Megaplex	\$14.5	40
3	Fairy Costume	DollarSmart	\$18	35
4	Thor Costume	Marvel Inc	\$23	55
5	Princess dress	Disney LTD	\$41	25

Rent another costume as well?  
Please enter 'y' to rent more costume or any other key to stop.y

Figure 11: Asking user to rent more and entering y to rent more

Enter the ID of the costume: 2

+++++

Costume is available

+++++

Enter the quantity of costume: 5

The price is: 14.5

ID	Costume Name	Company Name	Rent Price	Quantity
1	Cop Costume	Cartmax Inc	\$15.5	45
2	Formal Suite	Megaplex	\$14.5	35
3	Fairy Costume	DollarSmart	\$18	35
4	Thor Costume	Marvel Inc	\$23	55
5	Princess dress	Disney LTD	\$41	25

Rent another costume as well?  
Please enter 'y' to rent more costume or any other key to stop.n

Figure 12: Entering id and quantity and entering n to complete the process

```

=====
                        Bill Details
=====
*****

Name of customer: mike
Date Time of rent: 2022-08-26 13:40:38.767074

SN          Costume Name          Brand          Unit Price    Quantity
-----
1           Cop Costume           Cartmax Inc     15.5           4
2           Formal Suite          Megaplex        14.5           5
-----

Total price: 134.5

*****

|      Rent bill has been generated.      |
|      Your bill no is: 3071              |
|-----|

**  Note: If the costumes are not returned within 5 days,  **
**          there will be additional fine of $7.5 per day   **

```

Figure 13: Displaying the rent bill

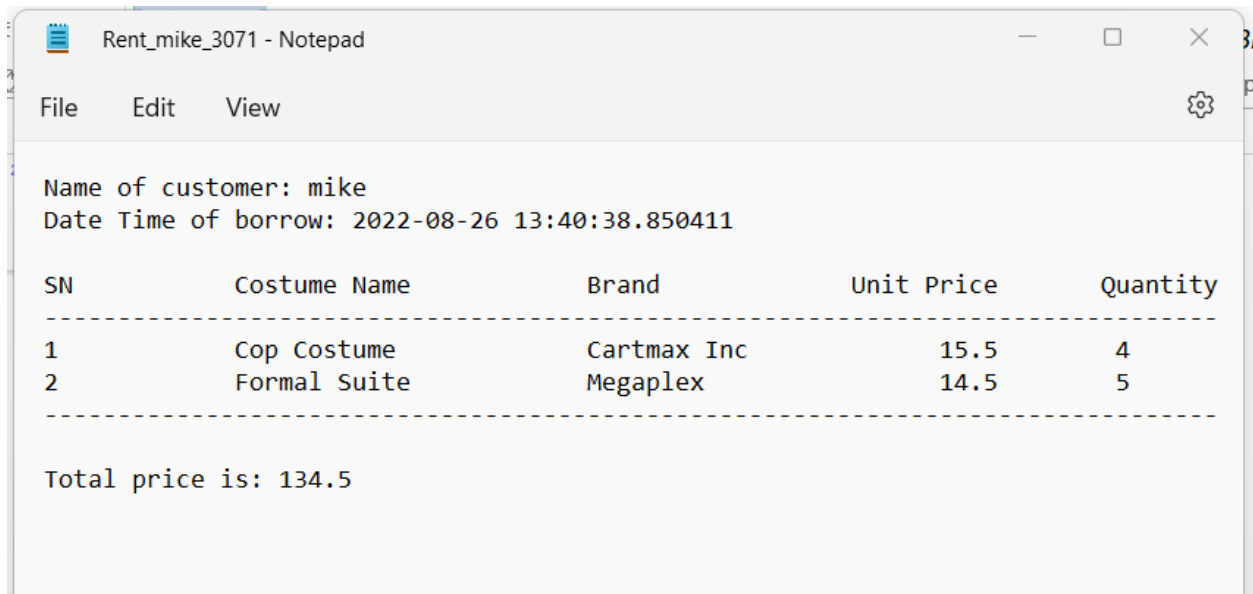


Figure 14: Rent bill in text file

### 3.2. Option2: Returning process

For the returning process, the user should enter Option2. Also, a user cannot return the costumes without the bill no from their rent bill so returning is not possible without renting the costumes first. When Option2 is entered, the user is asked to enter their name. The user can enter 'back' to go back to the options. After entering the name, they are asked the bill no and the number of renting days. Then the program checks for a rent bill with the input name and bill no. If the bill does not exist, the user is asked to re-enter the name and bill no. But if the bill exists, the costumes listed in the bill is returned and a bill for returning is displayed. A fine is also displayed if the number of renting days exceeds 5 days. At the same time, a text file for the bill is also generated.

Screenshots for returning process:

```

Please select an option:
(1) || Press 1 to rent costumes.
(2) || Press 2 to return costumes.
(3) || Press 3 to exit.

Enter your option: 2

+++++
Let's return a costume. You can enter 'back' as name to go back.
+++++

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume       Cartmax Inc       $15.5           45
2       Formal Suite         Megaplex          $14.5           35
3       Fairy Costume        DollarSmart       $18             35
4       Thor Costume         Marvel Inc        $23             55
5       Princess dress       Disney LTD        $41             25
-----

```

Figure 15: Enter option2 for returning

ID	Costume Name	Company Name	Rent Price	Quantity
1	Cop Costume	Cartmax Inc	\$15.5	45
2	Formal Suite	Megaplex	\$14.5	35
3	Fairy Costume	DollarSmart	\$18	35
4	Thor Costume	Marvel Inc	\$23	55
5	Princess dress	Disney LTD	\$41	25

Enter your name: mike  
Enter your bill no: 3071

Enter the no of renting days: 9

Figure 16: Entering name, bill no and rent days

```
=====
                        Bill Details
=====
*****

Name of customer: mike
Date Time of return: 2022-08-26 13:43:36.858746
No of renting days: 9


SN          Costume Name          Brand          Unit Price    Quantity
-----
1           Cop Costume            Cartmax Inc     15.5           4
2           Formal Suite            Megaplex        14.5           5
-----

Total price: 134.5

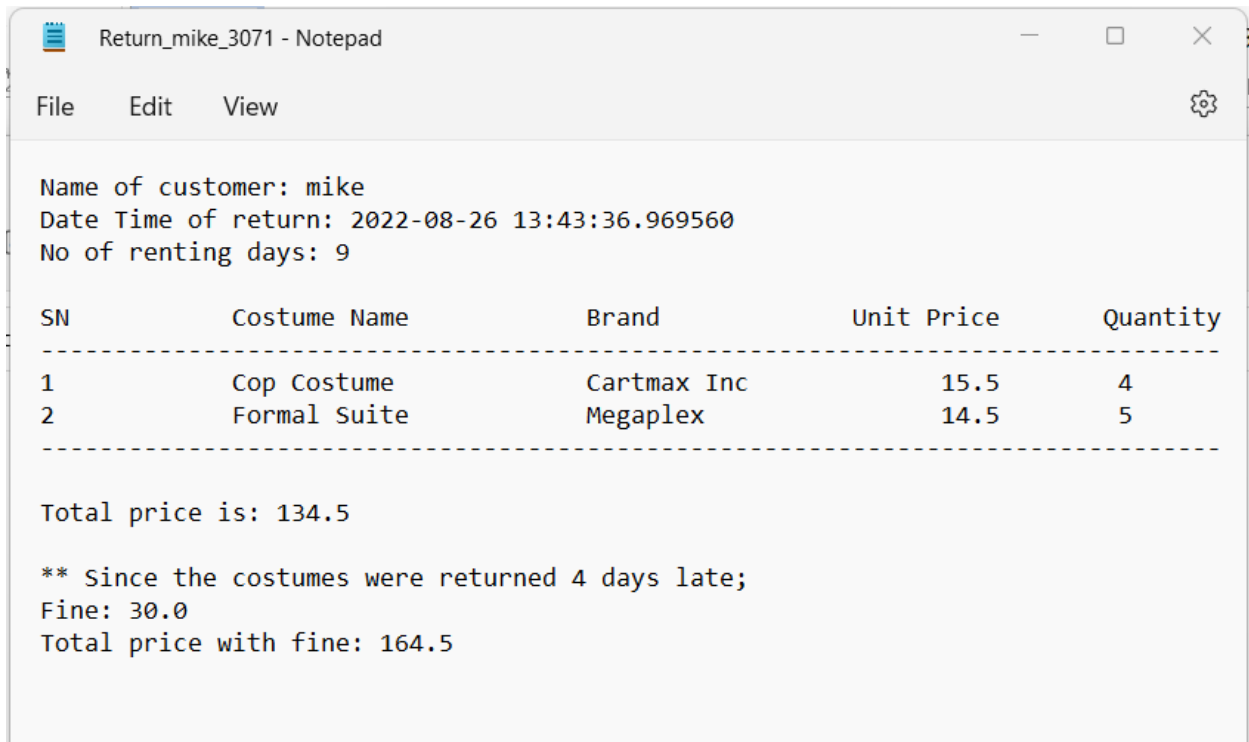
** Since the costumes were returned 4 days late;
Fine: 30.0
Total price with fine: 164.5

*****

-----
|           Return bill has been generated.           |
-----
```

Figure 17: Displaying the return bill with fine





```
Return_mike_3071 - Notepad
File Edit View
Name of customer: mike
Date Time of return: 2022-08-26 13:43:36.969560
No of renting days: 9

SN          Costume Name      Brand          Unit Price      Quantity
-----
1           Cop Costume         Cartmax Inc    15.5            4
2           Formal Suite             Megaplex       14.5            5
-----

Total price is: 134.5

** Since the costumes were returned 4 days late;
Fine: 30.0
Total price with fine: 164.5
```

Figure 18: Return bill as text file with fine

### 3.3. Option3: Exit

When the user enters Option3 as input, a thank you message is displayed and then the program terminates.

```
Please select an option:  
(1) || Press 1 to rent costumes.  
(2) || Press 2 to return costumes.  
(3) || Press 3 to exit.  
  
Enter your option: 3  
  
Thank you for using the application.
```

*Figure 19: Entering option3 and the program terminates*

## 4. Testing

### 4.1. Test 1

To show the implementation of try and except in the program.

Test no	1
Objective	To show the implementation of try and except in the program.
Action	<ul style="list-style-type: none"> <li>➤ The program was run in IDLE.</li> <li>➤ '1' was entered as the option and 'name' was entered as a name</li> <li>➤ For the id, a random string 'asd' was entered.</li> </ul>
Expected result	An error message is to be displayed for entering string value as id.
Actual result	The error message was shown.
Conclusion	The test was successful.

Table 1: Test1; implementation of try, except

### Screenshots for Test 1:

```
>>>
= RESTART: C:\Users\user\assign\Fundamental of computing\courseWork\weekly\main.py

-----
Welcome to Costume Rental Application
-----

Please select an option:
(1) || Press 1 to rent costumes.
(2) || Press 2 to return costumes.
(3) || Press 3 to exit.

Enter your option: 1

+++++
Let's rent a costume. You can enter 'back' as name to go back.
+++++

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume          Cartmax Inc       $15.5           49
2       Formal Suite          Megaplex          $14.5           40
3       Fairy Costume         DollarSmart       $18              35
4       Thor Costume          Marvel Inc        $23              55
5       Princess dress        Disney LTD        $41              25

Enter your name: name
```

Figure 20: Entering the option and name

Enter your option: 1

++++  
 Let's rent a costume. You can enter 'back' as name to go back.  
 +++++

ID	Costume Name	Company Name	Rent Price	Quantity
1	Cop Costume	Cartmax Inc	\$15.5	49
2	Formal Suite	Megaplex	\$14.5	40
3	Fairy Costume	DollarSmart	\$18	35
4	Thor Costume	Marvel Inc	\$23	55
5	Princess dress	Disney LTD	\$41	25

Enter your name: name

Enter the ID of the costume: asd

Figure 21: Entering string as id

Enter your name: name

Enter the ID of the costume: asd

++++  
 Invalid value for ID! Please enter a number.  
 +++++

Enter the ID of the costume:

Figure 22: Error message was shown

#### 4.2. Test 2

To select renting and returning of costumes and provide negative or non-existing values.

Test no	2
Objective	To select renting and returning of costumes and provide negative or non-existing values.
Action	<ul style="list-style-type: none"><li>➤ The program was run in IDLE.</li><li>➤ '1' was entered as the option for renting and 'ram' was entered as name.</li><li>➤ For the id, a negative value '-5' was entered.</li><li>➤ Then, a non-existing value '56' was entered</li></ul>
Expected result	Error messages are to be shown when entering negative and non-existing value.
Actual result	The error messages were shown.
Conclusion	The test was successful.

Table 2: Test 2; providing negative or non-existing values

## Screenshots for Test 2:

```
= RESTART: C:\Users\user\assign\Fundamental of computing\courseWork\weekly\main.py

-----
Welcome to Costume Rental Application
-----

Please select an option:
(1) || Press 1 to rent costumes.
(2) || Press 2 to return costumes.
(3) || Press 3 to exit.

Enter your option: 1

+++++
Let's rent a costume. You can enter 'back' as name to go back.
+++++

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume         Cartmax Inc       $15.5           49
2       Formal Suite          Megaplex          $14.5           40
3       Fairy Costume         DollarSmart       $18              35
4       Thor Costume          Marvel Inc        $23              55
5       Princess dress        Disney LTD        $41              25

Enter your name: ram
```

Ln: 41 Col: 20

Figure 23: Entering the option and name

```

Enter your option: 1

+++++
Let's rent a costume. You can enter 'back' as name to go back.
+++++

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume       Cartmax Inc       $15.5           49
2       Formal Suite        Megaplex          $14.5           40
3       Fairy Costume       DollarSmart       $18             35
4       Thor Costume         Marvel Inc        $23             55
5       Princess dress       Disney LTD        $41             25

Enter your name: ram

Enter the ID of the costume: -5

The input Costume ID in invalid! Please enter a valid ID.

Enter the ID of the costume:

```

Figure 24: Entering negative value

```

Enter your name: ram

Enter the ID of the costume: -5

The input Costume ID in invalid! Please enter a valid ID.

Enter the ID of the costume: 56

The input Costume ID in invalid! Please enter a valid ID.

Enter the ID of the costume:

```

Figure 25: Entering non-existing value

## 4.3. Test 3

To show complete renting process and the creation of the text file.

Test no	3
Objective	To show complete renting process and the creation of the text file.
Action	<ul style="list-style-type: none"> <li>➤ The program was run in IDLE.</li> <li>➤ '1' was entered as the option for renting and 'Hari' was entered as name.</li> <li>➤ '1' and '5' were entered for id and quantity.</li> <li>➤ Then 'y' was entered to rent more costumes.</li> <li>➤ '2' and '3' were entered for id and quantity for the second time.</li> <li>➤ Then 'n' was entered to display the bill.</li> </ul>
Expected result	The costumes are to be rented out while displaying the bill and also generating a text file.
Actual result	The costumes were rented out and bill was generated.
Conclusion	The test was successful.

Table 3 Test 3; complete renting process:



## Screenshots for Test 3:

```

-----
Welcome to Costume Rental Application
-----

Please select an option:
(1) || Press 1 to rent costumes.
(2) || Press 2 to return costumes.
(3) || Press 3 to exit.

Enter your option: 1

+++++
Let's rent a costume. You can enter 'back' as name to go back.
+++++

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume        Cartmax Inc       $15.5           49
2       Formal Suite          Megaplex          $14.5           40
3       Fairy Costume         DollarSmart       $18             35
4       Thor Costume          Marvel Inc        $23             55
5       Princess dress        Disney LTD        $41             25

Enter your name: Hari

```

Figure 26: Entering option and name for renting

```

Enter your name: Hari

Enter the ID of the costume: 1

+++++
Costume is available
+++++

Enter the quantity of costume: 5

```

Figure 27: Entering id and quantity

```

+++++
Costume is available
+++++

Enter the quantity of costume: 5

The price is: 15.5

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume       Cartmax Inc       $15.5           44
2       Formal Suite        Megaplex          $14.5           40
3       Fairy Costume       DollarSmart       $18             35
4       Thor Costume         Marvel Inc        $23             55
5       Princess dress       Disney LTD        $41             25

Rent another costume as well?
Please enter 'y' to rent more costume or any other key to stop.y

```

Figure 28: Choosing to rent more

```

Rent another costume as well?
Please enter 'y' to rent more costume or any other key to stop.y

Enter the ID of the costume: 2

+++++
Costume is available
+++++

Enter the quantity of costume: 3

```

Figure 29: Entering the id and quantity second time

```
+++++
Costume is available
+++++
```

```
Enter the quantity of costume: 3
```

```
The price is: 14.5
```

ID	Costume Name	Company Name	Rent Price	Quantity
1	Cop Costume	Cartmax Inc	\$15.5	44
2	Formal Suite	Megaplex	\$14.5	37
3	Fairy Costume	DollarSmart	\$18	35
4	Thor Costume	Marvel Inc	\$23	55
5	Princess dress	Disney LTD	\$41	25

```
Rent another costume as well?
```

```
Please enter 'y' to rent more costume or any other key to stop.n
```

Figure 30: Completing the renting process

```
=====
                        Bill Details
=====
*****

Name of customer: Hari
Date Time of rent: 2022-08-26 09:19:04.256061

SN          Costume Name          Brand          Unit Price    Quantity
-----
1           Cop Costume           Cartmax Inc     15.5          5
2           Formal Suite          Megaplex        14.5          3
-----

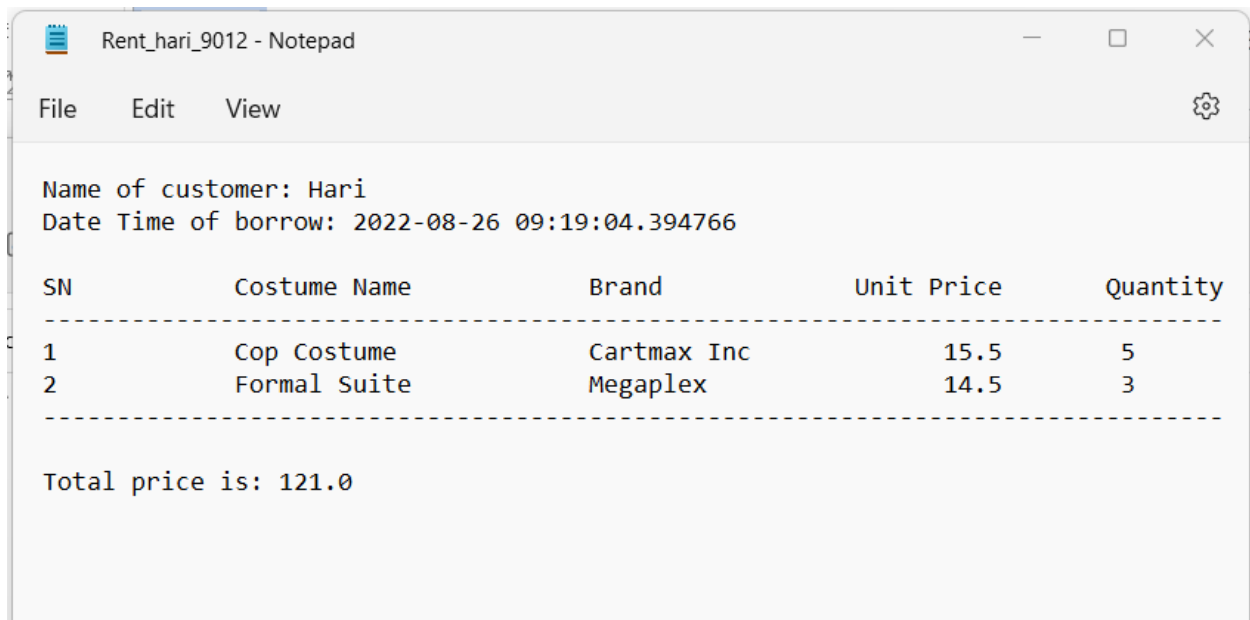
Total price: 121.0

*****

-----
|           Rent bill has been generated.           |
|           Your bill no is: 9012                    |
-----

**    Note: If the costumes are not returned within 5 days,    **
**          there will be additional fine of $7.5 per day      **
```

Figure 31: Rent bill being displayed



```

Rent_hari_9012 - Notepad
File Edit View
Name of customer: Hari
Date Time of borrow: 2022-08-26 09:19:04.394766
SN          Costume Name      Brand          Unit Price    Quantity
-----
1           Cop Costume        Cartmax Inc    15.5          5
2           Formal Suite        Megaplex       14.5          3
-----
Total price is: 121.0

```

Figure 32: Rent bill in text file

## 4.4. Test 4

To show complete returning process and the creation of text file.

Test no	4
Objective	To show complete returning process and the creation of text file.
Action	<ul style="list-style-type: none"> <li>➤ The program was run in IDLE.</li> <li>➤ '2' was entered as the option for returning.</li> <li>➤ The name and bill no 'Hari' and '9012' from the previous renting bill were entered.</li> <li>➤ '7' was entered for the no. of renting days and then the bill was displayed.</li> </ul>
Expected result	The rented costumes are to be returned and bill details are to be displayed along with creation of text file.
Actual result	The costumes were returned and the bill for renting was generated.
Conclusion	The test was successful.

Table 4: Test4; complete returning process

## Screenshots for Test 4:

```
>
===== RESTART: C:\Users\user\assign\Fundamental of computing\courseWork\weekly\main.py =====
-----
                        Welcome to Costume Rental Application
-----

Please select an option:
(1) || Press 1 to rent costumes.
(2) || Press 2 to return costumes.
(3) || Press 3 to exit.

Enter your option: 2
```

Figure 33: Choosing option to return

```

Enter your option: 2

+++++
Let's return a costume. You can enter 'back' as name to go back.
+++++

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume       Cartmax Inc       $15.5           44
2       Formal Suite        Megaplex          $14.5           37
3       Fairy Costume       DollarSmart       $18             35
4       Thor Costume         Marvel Inc        $23             55
5       Princess dress        Disney LTD        $41             25

Enter your name: Hari
Enter your bill no: 9012

```

Figure 34: Entering name and bill no

```

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume       Cartmax Inc       $15.5           44
2       Formal Suite        Megaplex          $14.5           37
3       Fairy Costume       DollarSmart       $18             35
4       Thor Costume         Marvel Inc        $23             55
5       Princess dress        Disney LTD        $41             25

Enter your name: Hari
Enter your bill no: 9012

Enter the no of renting days: 7

```

Figure 35: Entering no. of days

```
=====
                        Bill Details
=====
*****

Name of customer: Hari
Date Time of return: 2022-08-26 09:22:26.500463
No of renting days: 7

SN          Costume Name          Brand          Unit Price    Quantity
-----
1           Cop Costume           Cartmax Inc     15.5           5
2           Formal Suite          Megaplex        14.5           3
-----

Total price: 121.0

** Since the costumes were returned 2 days late;
Fine: 15.0
Total price with fine: 136.0

*****

-----
|           Return bill has been generated.           |
-----
```

Figure 36: Displaying the bill for return with fine



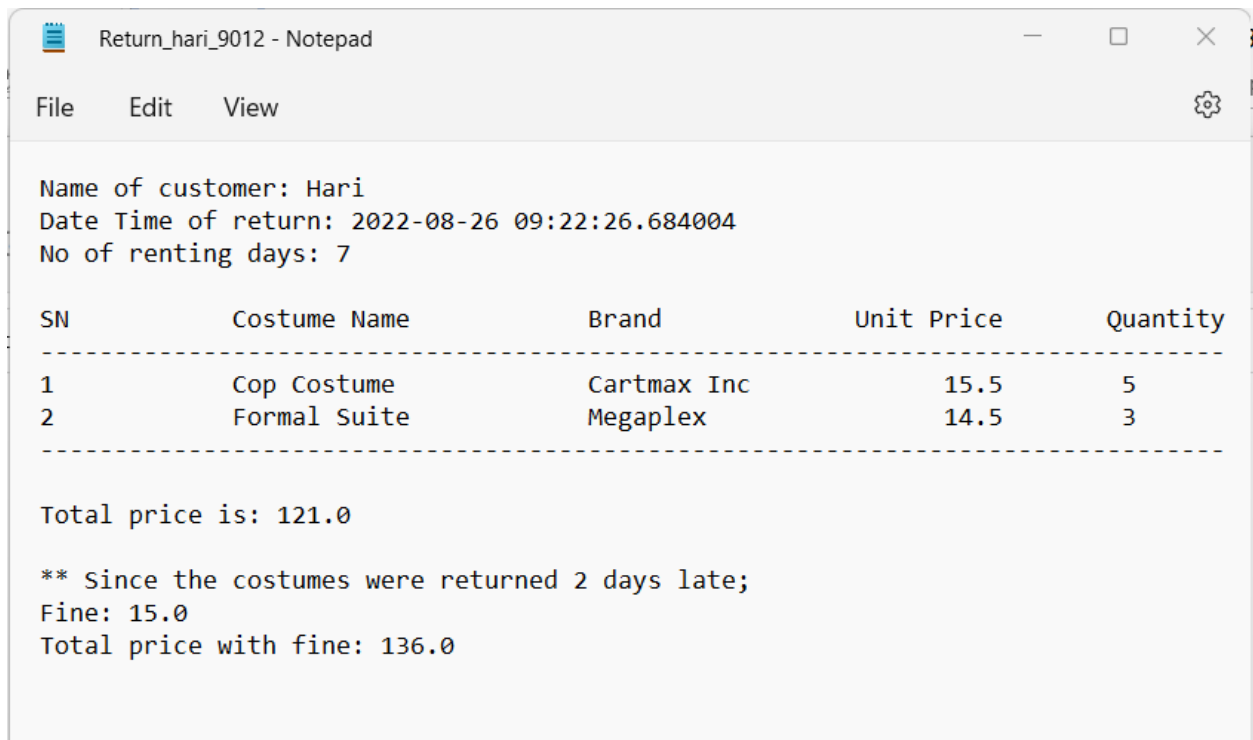


Figure 37: Bill for return in text file

## 4.5. Test 5

To show the deduction and addition of quantity of costumes when they are rented and returned.

Test no	5
Objective	To show the deduction and addition of quantity of costumes when they are rented and returned.
Action	<ul style="list-style-type: none"> <li>➤ The program was run in IDLE.</li> <li>➤ '1' was entered as the option for renting and 'Andy' was entered as name.</li> <li>➤ '5' quantity of costume id '2' and then '10' quantity of another id '3' were rented.</li> <li>➤ Then the updated costume details were displayed.</li> <li>➤ After renting, '2' was entered as option for returning.</li> <li>➤ The name 'Andy' and bill no '2674' from the previous rent bill was entered.</li> <li>➤ Then the updated costume details was displayed.</li> </ul>
Expected result	The quantity should be deducted after renting and should be increased after returning.
Actual result	The quantity was deducted after renting and increased after returning.
Conclusion	The test was successful.

Table 5: Test5; showing change in quantity after renting and returning

## Screenshots for Test 5:

```

Please select an option:
(1) || Press 1 to rent costumes.
(2) || Press 2 to return costumes.
(3) || Press 3 to exit.

Enter your option: 1

+++++
Let's rent a costume. You can enter 'back' as name to go back.
+++++

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume        Cartmax Inc       $15.5           49
2       Formal Suite          Megaplex          $14.5           40
3       Fairy Costume         DollarSmart       $18             35
4       Thor Costume          Marvel Inc        $23             55
5       Princess dress        Disney LTD        $41             25

Enter your name: Andy

```

Figure 38: Entering option for renting and name

```

Enter your name: Andy

Enter the ID of the costume: 2

+++++
Costume is available
+++++

Enter the quantity of costume: 5

```

Figure 39: Entering the id and quantity

Enter the quantity of costume: 5

The price is: 14.5

ID	Costume Name	Company Name	Rent Price	Quantity
1	Cop Costume	Cartmax Inc	\$15.5	49
2	Formal Suite	Megaplex	\$14.5	35
3	Fairy Costume	DollarSmart	\$18	35
4	Thor Costume	Marvel Inc	\$23	55
5	Princess dress	Disney LTD	\$41	25

Rent another costume as well?

Please enter 'y' to rent more costume or any other key to stop.y

Figure 40: Choosing to rent more

Rent another costume as well?

Please enter 'y' to rent more costume or any other key to stop.y

Enter the ID of the costume: 3

++++  
Costume is available  
++++

Enter the quantity of costume: 10

Figure 41: Entering id and quantity again

ID	Costume Name	Company Name	Rent Price	Quantity
1	Cop Costume	Cartmax Inc	\$15.5	49
2	Formal Suite	Megaplex	\$14.5	40
3	Fairy Costume	DollarSmart	\$18	35
4	Thor Costume	Marvel Inc	\$23	55
5	Princess dress	Disney LTD	\$41	25

Figure 42: Original costume details

ID	Costume Name	Company Name	Rent Price	Quantity
1	Cop Costume	Cartmax Inc	\$15.5	49
2	Formal Suite	Megaplex	\$14.5	35
3	Fairy Costume	DollarSmart	\$18	25
4	Thor Costume	Marvel Inc	\$23	55
5	Princess dress	Disney LTD	\$41	25

Figure 43: Costume details after renting

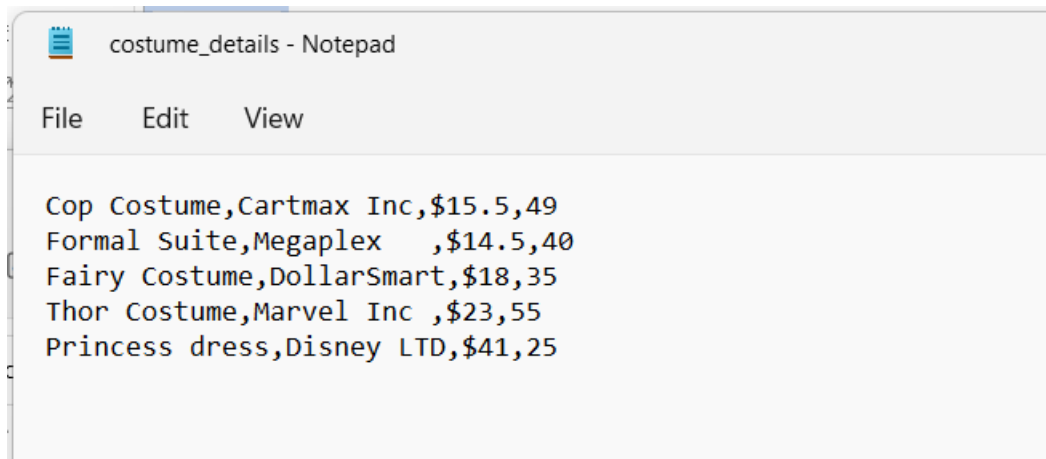


Figure 44: Original costume details in text file

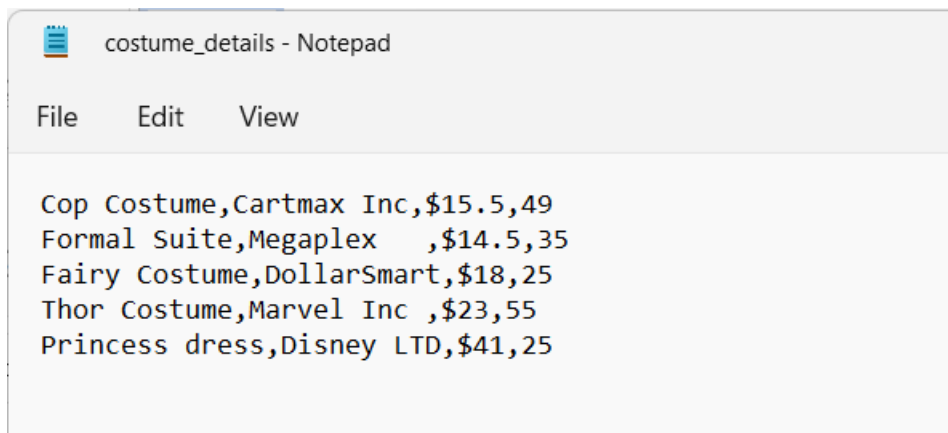


Figure 45: Costume details in text file after renting

```

Please select an option:
(1) || Press 1 to rent costumes.
(2) || Press 2 to return costumes.
(3) || Press 3 to exit.

Enter your option: 2

```

Figure 46: Entering the option for returning

```

Enter your option: 2

+++++
Let's return a costume. You can enter 'back' as name to go back.
+++++

-----
ID      Costume Name      Company Name      Rent Price      Quantity
-----
1       Cop Costume       Cartmax Inc       $15.5           49
2       Formal Suite        Megaplex          $14.5           35
3       Fairy Costume        DollarSmart       $18             25
4       Thor Costume         Marvel Inc        $23             55
5       Princess dress       Disney LTD        $41             25

Enter your name: Andy
Enter your bill no: 2674

Enter the no of renting days: 5

```

Figure 47: Entering name, bill no and no. of days

ID	Costume Name	Company Name	Rent Price	Quantity
1	Cop Costume	Cartmax Inc	\$15.5	49
2	Formal Suite	Megaplex	\$14.5	35
3	Fairy Costume	DollarSmart	\$18	25
4	Thor Costume	Marvel Inc	\$23	55
5	Princess dress	Disney LTD	\$41	25

Figure 48: Costume details before returning

ID	Costume Name	Company Name	Rent Price	Quantity
1	Cop Costume	Cartmax Inc	\$15.5	49
2	Formal Suite	Megaplex	\$14.5	40
3	Fairy Costume	DollarSmart	\$18	35
4	Thor Costume	Marvel Inc	\$23	55
5	Princess dress	Disney LTD	\$41	25

Figure 49: Costume details after returning, back to original



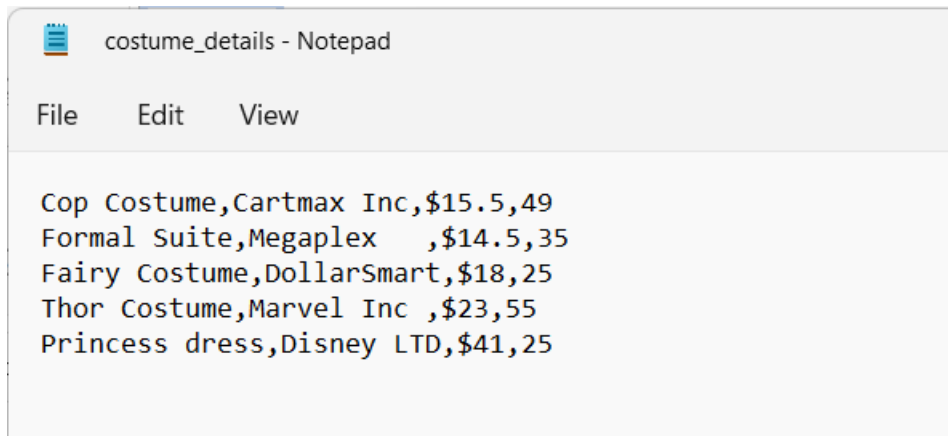


Figure 50: Costume details before returning in text file

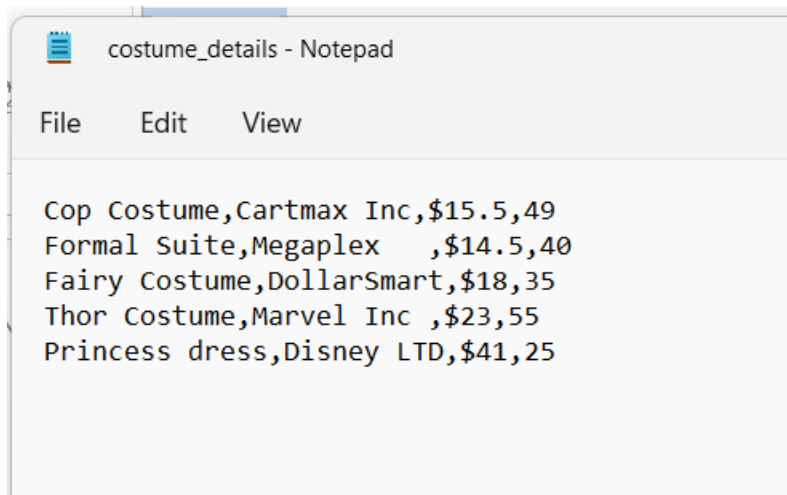


Figure 51: Costume details after returning in text file

## 5. Conclusion

During the development of the program, various problems had emerged and the logical errors in the program were the most frequent ones. For example, the error while displaying the bill the list of costumes wouldn't be displayed or error while calculating fine for returning. Similarly, various errors were also encountered while writing and reading from a text file. Likewise, iterating through the dictionaries and using the built-in functions like `split()` and `replace()` were also quite hard to understand and implement into the program in the beginning. In the end, all these problems were solved quite easily through more practice, consulting the tutors and a little research in the Internet.

Since various problems were faced and solved throughout the development of this project, this has become a great learning experience. This project has helped in understanding the various data structures in python and when each of them should be used and should not be used. Also the process iterating through these data structures to manipulate the values has become clearer by the end of this project. This project has also helped in understanding how one can read from and write into a text file as well as how these extracted data can be stored and manipulated. Lastly, this project has also increased the knowledge about how loops work and how one can use it to create a smoother execution of a program for the users.

## References

CFI Team, 2021. *Python Data Structures*. [Online]  
Available at: <https://corporatefinanceinstitute.com/resources/knowledge/other/python-data-structures/>

[Accessed 25 08 2022].

GeekforGeeks, 2022. *Python Data Structures*. [Online]  
Available at: <https://www.geeksforgeeks.org/python-data-structures/>

[Accessed 25 08 2022].

Loshin, D., 2021. *What are data structures?*. [Online]  
Available at: <https://www.techtarget.com/searchdatamanagement/definition/data-structure>

[Accessed 25 08 2022].

W3Schools, 2018. *Python Tutorial*. [Online]  
Available at: <https://www.w3schools.com/python/default.asp>

[Accessed 20 08 2022].

## Appendix

- Code for main.py

```
from operations import options,option_1,option_2,option_3,invalid_option
```

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

```
print("          Welcome to Costume Rental Application")
```

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

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

```
run = True
```

```
while run:
```

```
    options()
```

```
    user_input = input("Enter your option: ")
```

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

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

```
        option_1()#calls the function for renting costumes
```

```
    elif user_input == "2":
```

```
        option_2()#calls the function for renting costumes
```

```
    elif user_input == "3":
```

```
        option_3()#for exiting the program
```

```
        run = False
```

```
    else:
```

```
        invalid_option()#displays invalid message when wrong input is entered
```

- Code for operations.py

```
import datetime,random
```

```
import return_operations
```

```
def costumes():
```

```
    """displays the details of the costumes"""
```

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

```
    print("ID      Costume Name      Company Name      Rent Price      Quantity")
```

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

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

```
    id = 1
```

```
    #reading from file and printing the costume details
```

```
    for line in file:
```

```
        print(id,"\t"+line.replace(",","\t\t"))
```

```
        id += 1
```

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

```
    file.close()
```

```
def dictionary_creation():
```

```
    """stores the costume details in a dictionary and returns it"""
```

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

```
    costume_dictionary = {}
```

```
    costume_id = 1
```

```
    #reading from the text file and storing the details in a dictionary
```

```
    for line in file:
```

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

```
        costume_dictionary.update({costume_id:(line.split(","))})
```

```
        costume_id += 1
```

```
    file.close()
```

```
    return costume_dictionary
```

```

def text_update(costume_dictionary):
    """updates the text file"""
    file = open("costume_details.txt","w")
    #updating the text file after renting costumes
    for key,value in costume_dictionary.items():
        file.write(value[0] + "," + value[1] + "," + value[2] + "," + value[3])
        file.write("\n")
    file.close()

def quantity_update(input_id,quantity):
    """updates dictionary after renting"""
    costume_dictionary = dictionary_creation()
    available_quantity = int(costume_dictionary[input_id][3])
    while quantity<=0 or quantity>available_quantity: #to check the input quantity is
available
        if quantity>available_quantity:
            print("Quantity provided is greater than what we have in stock.")
        else:
            print("Enter a valid quantity.")
        print("\n")
        success = False
        while success == False:
            try:
                quantity = int(input("Enter the quantity of costume: "))
                success = True
            except:
                #exception when invalid input is given as quantity
                print("\n")
                print("+++++")
                print("Invalid value for quantity! Please enter a number.")
                print("+++++")

```

```

        print("\n")

    print("\n")
    costume_dictionary[input_id][3] = str(available_quantity - quantity)#updating quantity
    in dictionary
    text_update(costume_dictionary)#calling function to update the text file
    return costume_dictionary,quantity

def generate_bill(name,total_price,item_list,bill_no):
    """creates the bill in text file"""
    SN = 0
    file = open("Rent_"+name.lower()+"_"+bill_no+".txt","w")
    file.write("Name of customer: "+name)
    file.write("\n")
    file.write("Date Time of borrow: "+str(datetime.datetime.now()))
    file.write("\n\n")
    file.write("SN \t\t Costume Name \t\t Brand \t\t Unit Price \tQuantity")
    file.write("\n")
    file.write("-----")
    file.write("\n")
    for each in item_list: #writing the list of costumes along with brand, price and quantity
        into txt file
        SN += 1
        file.write(str(SN)+"\t\t "+each[0)+"\t\t "+each[1)+"\t\t "+str(each[2])+"\t\t "+str(each[3]))
        file.write("\n")
    file.write("-----")
    file.write("\n\n")
    file.write("Total price is: "+str(total_price))
    file.write("\n")
    file.close()

```

```

def bill(name,total_price,item_list):
    """displays the bill"""
    SN = 0
    bill_no = str(random.randint(0,10000))# generates random number
    print("=====")
    print("          Bill Details")
    print("=====")
    print("*****")
    print("\n")
    print("Name of customer:",name)
    print("Date Time of rent:",datetime.datetime.now())
    print("\n")
    print("SN \t Costume Name \t\t Brand \t\t Unit Price \t Quantity")
    print("-----")
    for each in item_list: #printing the list of costumes along with brand, price and quantity
        SN += 1
        print(str(SN)+"\t"+each[0]+\t\t+each[1]+\t  "+str(each[2])+"\t\t "+str(each[3]))
        print("\n")
    print("-----")
    print("\n")
    print("Total price:",total_price)
    print("\n")
    print("*****")
    print("\n")
    generate_bill(name,total_price,item_list,bill_no)#calls the function to generate text bill
    print("-----")
    print("|      Rent bill has been generated.      |")
    print("|      Your bill no is:",bill_no,"          |")
    print("-----")
    print("\n")

```



```

print("*** Note: If the costumes are not returned within 5 days,  **")
print("***      there will be additional fine of $7.5 per day      **")
print("\n")

```

```
def valid_id():
```

```
    """to check if input id is valid and returns the valid id """
```

```
    success = False
```

```
    while success == False:
```

```
        try:
```

```
            input_id = int(input("Enter the ID of the costume: "))
```

```
            success = True
```

```
        except:
```

```
            #exception when the input id is invalid
```

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

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

```
            print("Invalid value for ID! Please enter a number.")
```

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

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

```
    while input_id <= 0 or input_id > len(dictionary_creation()):# to check if the id exists in
the dictionary
```

```
        print("\n The input Costume ID in invalid! Please enter a valid ID. \n")
```

```
        success = False
```

```
        while success == False:
```

```
            try:
```

```
                input_id = int(input("Enter the ID of the costume: "))
```

```
                success = True
```

```
            except:
```

```
                #exception when the input id is invalid
```

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

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

```
                print("Invalid value for ID! Please enter a number.")
```

```

        print("+++++")
        print("\n")
    print("\n")
    print("+++++")
    print("Costume is available")
    print("+++++")
    print("\n")
    return input_id

```

```

def renting(name):
    """for renting costumes"""
    rent_more = True
    item_list = []
    SN = 0
    total_price = 0
    while rent_more:#loops as long as the user wants to rent more
        input_id = valid_id()#calls the function to check if input id is valid and stores the valid
id
        success = False
        while success == False:
            try:
                quantity = int(input("Enter the quantity of costume: "))
                success = True
            except:
                #exception when the input quantity is invalid
                print("\n")
                print("+++++")
                print("Invalid value for quantity! Please enter a number.")
                print("+++++")
                print("\n")
        print("\n")

```

```

    updating_dictionary = quantity_update(input_id,quantity)#calls the function to
update quantity in dictionary

```

```

    costume_dictionary = updating_dictionary[0]
    quantity = updating_dictionary[1]
    price = float((costume_dictionary[input_id][2]).replace("$",""))
    total_price += price*quantity
    print("The price is:",price)
    print("\n")

```

```

item_list.append([costume_dictionary[input_id][0],costume_dictionary[input_id][1],price,
quantity])

```

```

    costumes()
    print("Rent another costume as well?")
    more = input("Please enter 'y' to rent more costume or any other key to stop.")
    print("\n")
    if more.lower() != "y":#asking user if they want to rent more costumes
        rent_more = False
    bill(name,total_price,item_list)#calls the function to display the bill

```

```

def options():

```

```

    """displays the options to the user"""
    print("Please select an option:")
    print("(1) || Press 1 to rent costumes.")
    print("(2) || Press 2 to return costumes.")
    print("(3) || Press 3 to exit.")
    print("\n")

```

```

def option_1():

```

```

    """when renting costumes"""

```

```

print("+++++
++")
    print(" Let's rent a costume. You can enter 'back' as name to go back.")

print("+++++
++")
    print("\n")
    costumes()#to display the available costumes
    print("\n")
    name = input("Enter your name: ")
    print("\n")
    if name.lower() != "back":
        renting(name)

def option_2():
    """for returning costumes"""

print("+++++
++++")
    print(" Let's return a costume. You can enter 'back' as name to go back.")

print("+++++
++++")
    print("\n")
    costumes()#to display the available costumes
    return_operations.returning()#calls the function to return costumes

def option_3():
    """to exit the application"""
    print("      Thank you for using the application.")

```

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

```
def invalid_option():
```

```
    """displays error when invalid option is input"""
```

```
    print("Invalid Option!!!!")
```

```
    print("Please select the value as per the given options.")
```

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

- Code for return\_operations.py

```
import datetime
```

```
import operations
```

```
def ask_rent_days():
```

```
    """asks the no of renting days to the user"""
```

```
    days_success = False
```

```
    while days_success == False:#to check if the input days is valid or not
```

```
        try:
```

```
            positive = False
```

```
            while positive == False:#to check if the input days is positive or not
```

```
                rent_days = int(input("Enter the no of renting days: "))
```

```
                if rent_days > 0:
```

```
                    positive = True
```

```
            else:
```

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

```
                print("Invalid value for no of renting days! Please enter a positive number.")
```

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

```
        days_success = True
```

```
    except:#exception when invalid input is given
```

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

```
        print("Invalid value for no of renting days! Please enter a number.")
```

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

```
    return rent_days
```

```
def return_bill_text(name,bill_no,record,total_price,rent_days,fine):
```

```
    """creates the bill of returning in text file"""
```

```
    file = open("Return_"+name.lower()+"_"+bill_no+".txt","w")
```

```
    file.write("Name of customer: "+name)
```

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

```

file.write("Date Time of return: "+str(datetime.datetime.now()))
file.write("\n")
file.write("No of renting days: "+str(rent_days))
file.write("\n\n")
file.write("SN \t\t Costume Name \t\t Brand \t\t Unit Price \tQuantity")
file.write("\n")
file.write("-----")
file.write("\n")
for each in record.values(): #writing the list of costumes along with brand, price and
quantity into txt file
    file.write(str(each[0])+"\t\t " +each[1]+"\t\t " +each[2]+"\t\t " +str(each[3])+"\t\t "
"+str(each[4]))
    file.write("\n")
file.write("-----")
file.write("\n\n")
file.write("Total price is: "+str(total_price))
file.write("\n")
if fine !=0: #writing the fine into txt file if returned late
    file.write("\n")
    file.write("*** Since the costumes were returned "+str(rent_days-5)+" days late;")
    file.write("\n")
    file.write("Fine: "+str(fine))
    file.write("\n")
    file.write("Total price with fine: "+str(fine+total_price))
    file.write("\n")
file.close()

def return_bill(name,bill_no,record,total_price,rent_days):
    """displays the bill of returning"""
    fine = 0
    print("=====")

```

```

print("          Bill Details")
print("=====")
print("*****")
print("\n")
print("Name of customer:",name)
print("Date Time of return:",datetime.datetime.now())
print("No of renting days:",rent_days)
print("\n")
print("SN \t Costume Name \t\t Brand \t\t Unit Price \t Quantity")
print("-----")
for each in record.values(): #printing the list of costumes along with brand, price and
quantity
    print(str(each[0])+"\t"+each[1]+"\t\t"+each[2]+"\t  "+str(each[3])+"\t\t",str(each[4]))
    print("\n")
print("-----")
print("\n")
print("Total price:",total_price)
if rent_days > 5:#to check if the costumes are returned within the timelimit
    fine = 7.5 * (rent_days - 5)
    print("\n")
    print("*** Since the costumes were returned",rent_days-5,"days late;")
    print("Fine:",fine)
    print("Total price with fine:",total_price + fine)
print("\n")
print("*****")
print("\n")
return_bill_text(name,bill_no,record,total_price,rent_days,fine)#calls the function to
generate text bill
print("-----")
print("|      Return bill has been generated.      |")
print("-----")

```



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

```
def return_update(record):
```

```
    """updates the costume dictionary when costume is returned"""
```

```
    costume_dictionary = operations.dictionary_creation()#calls the function to create
    dictionary from text file of costume details
```

```
    for value in costume_dictionary.values():
```

```
        for records in record.values():
```

```
            if value[0] == records[1]:
```

```
                value[3] = str(int(value[3])+int(records[4]))
```

```
    operations.text_update(costume_dictionary)# call the function to update the text file
```

```
def details_from_bill(name,bill_no):
```

```
    file = open("Rent_"+name.lower()+"_"+bill_no+".txt","r")
```

```
    iteration = 0
```

```
    lines = list(file)
```

```
    record = {}
```

```
    rID = 0
```

```
    total_price = 0
```

```
    for line in lines:#extracting the records from the rent bill and storing in a dictionary
```

```
        iteration += 1
```

```
        if iteration>5 and iteration<(len(lines)-2):
```

```
            rID += 1
```

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

```
            record.update({rID:line.split("\t\t ")})
```

```
        if iteration == len(lines):
```

```
            total_price = float(line.replace("Total price is: ",""))
```

```
    file.close()
```

```
    return record,total_price
```

```
def returning():
```

```
"""for returning costumes"""
success = False
while success == False:
    name = input("Enter your name: ")
    if name.lower() == "back":
        break
    bill_no = input("Enter your bill no: ")
    print("\n")
    try:
        file = open("Rent_"+name.lower()+"_"+bill_no+".txt", "r")
        file.close()
        success = True
    except:#exception when the bill of input name or bill no does not exist
        print("The input customer name or bill no is incorrect! Please enter the correct
value.")
        print("\n")
    if name.lower() != "back":
        bill_details = details_from_bill(name,bill_no)
        record = bill_details[0]
        total_price = bill_details[1]

    rent_days = ask_rent_days())#calls function to ask no of days
    return_update(record)#calling the function to update dictionary after returning
    return_bill(name,bill_no,record,total_price,rent_days)# calls function to generate bill
after returning
```

- Details in costume\_details.txt

Cop Costume, Cartmax Inc, \$15.5, 49

Formal Suite, Megaplex, \$14.5, 40

Fairy Costume, DollarSmart, \$18, 35

Thor Costume, Marvel Inc, \$23, 55

Princess dress, Disney LTD, \$41, 25