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 Summer

Student Name: Anju Kumari Yadav

Group: C15

LonDOn Met ID: 22015649

College ID: NP01CP4S220194

Assignment Due Date: 26th August 2022

Assignment Submission Date: 26th 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 for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

List of Figures

List of Tables

1. Introduction:

The requirement for costumes remains year-round for many clients, even if Halloween sees a spike in demand. Wigs, makeup, and costumes for theater performances, festivals, and special occasions are offered to customers by costume shops. Successful businesses offer a variety of items that can be rented and purchased depending on the needs of their customers.

A costume rental business is a lot of fun, and it may add significantly to your yearly income. The rental income is almost entirely profit after the first five or six rentals, when the item has paid for itself. The most effective way to market a costume rental business is to make a marketing brochure that lists the costumes that are currently in stock and distribute it along with a rental rate sheet to community organizations like sports teams, community theater companies, charity organizations, and colleges and universities. You'll make a lot of money around Halloween, so make sure to increase your marketing activity in September and October. Once established, this kind of rental business will be able to survive on repeat business and word-of-mouth marketing.

A straightforward project called the Costume Rental System was created to aid rental shops in managing costumes. Python programming is used to create a costume rental system that gives the business owner control over the rental and return of costumes. This project shows how to build a Costume Rental System user interface without utilizing any Python GUI toolkits. The output is displayed on the terminal, and the interface is simply built. The three different types of modules that make up the Costume Rental System are in charge of carrying out the system's operational tasks. Costume information, including name, brand, price, and quantity, may be found in the main module, rent Module, returnC Module, and costumes.txt file. The system also makes use of the concept of structures to specify the shop products. It also

effectively employs a number of Python concepts, such as file operations, looping, and functions for manipulating strings.

For each transaction, the rent and return module generates a note. The rent note and return note, which are generated when someone rents a costume, contain the name of the customer, the name of the costume rented with Brand name, cost, and quantity, as well as the date and time the costume was rented by the user. The letter also calculates the overall rental balance owing. The numerous costume renting condition is also implemented using Python loops and the conditional function of the rent module. In order to avoid confusion, if someone decides to rent many costumes, they should put each one on the note along with the total amount, which is also calculated on the note. When a costume is brought back to the shop, a note should be made and added to the file once more. In the note, the customer's name, the name and brand of the costume, the quantity of costumes being returned, the date, and the time of the return are all stated. If the costume is returned after the allotted 5 days, a daily $10 fine will be applied to the return letter.

Goals of Costume Rental System

This project's main objective was to develop a system for costume rentals that keeps outfit details in a text file. It was necessary to develop a program that can read text files and display every item of clothing that is rentable. Then, for each transaction involving renting and returning, a note for the specific borrower should be created and kept in a file. After each transaction, the costume supply should also be updated.

Some of the important points are listed below:

1. A program that reads the text file and displays all the costumes that are offered for rental must be created.
2. Then, for each rental transaction, a note should be created for that specific rental and stored in a file.

3. The quantity of the costumes should be reduced by the number of costumes the user has taken, and the stock of costumes should also be updated after each transaction in accordance with this quantity.
4. A note should be generated once more for the person returning the costume in the event that one is being returned.
5. Following the user's return, the stock has to be updated as well.
6. It is recommended that the costume stock receive more returned costumes.
7. The costume return period should not exceed five days.
8. If the borrower returns the costume after the due date, a fine of $10 per day should be assessed.

Objective of Costume Rental Shop

The project mainly focuses on the following objectives:

1. To develop a project utilizing the features of Python programming
2. To put into practice functions like control statements, structures, and exception handling
3. To build the user-friendly and simple application interface
4. To gain knowledge of how to create a straightforward Python project.
5. to get knowledge for tackling actual problem-based projects and learn how to create complex programs geared at solving real-life issues like the Costume Rental System.
6. The ability to create, compile, and debug python scripts in order to address problem-based difficulties.

Tools Used

- **MS-WORD**

  Microsoft Word is a commercial, non-free word processor Created by Microsoft. It was first launched in 1983 FOR Xeinx systems under the name Multi-Tool Word.

Later versions were built FOR a variety of platforms, including IBM PCs running DOS (1983), Apple Macintosh (1984), AT&T Unix PC (1985), Atari ST (1986), and Microsoft Windows (1987). (1989). It is part of the Microsoft Office system, but it is also available as a standalone application and as part of the Microsoft Works Suite. Microsoft Word 2010 FOR Windows and 2011 FOR Mac are the most recent versions. Microsoft Word is a processor that can CREATE simple and complex writings. The application can be Downloaded to your hard drive or used online. You may share and collaborate on your files with others in real time with the online version. The application supports Windows, macOS, cell phones, and tablets.

- **Draw.io**

  Draw.io is a free online diagramming tool that lets you CREATE charts and diagrams. You can either CREATE a custom layout with the application or use the automated layout option. They have many shapes and hundreds of them. visual elements that can be used to CREATE diagrams and charts. The drag-and-drop feature makes constructing a professional-looking diagram or chart a breeze. A piece of cake We can CREATE flowcharts, UML diagrams, and entity relationship diagrams FOR you. Mock-ups, schematics, network diagrams, and more the draw.io application saves data. Google Drive, OneDrive, and other options are available through the program. Others include Dropbox, GitHub, and GitLab.

- **Python IDLE**

  IDLE is python's Integrated Development and Learning Environment. The Python installer FOR Windows contains the IDLE module by default. It allow programmer to easily write python code. Just like Python Shell, IDLE can be used to execute a single statement and CREATE, modify and execute Python scripts. IDLE provides a fully featured text editor to CREATE Python scripts that includes

feature like syntax highlighting, autocompletion, and smart indent. It also has a debugger with stepping and breakpoints features.

2. Discussion and Analysis:

A. Algorithm

In computer programming terms, an algorithm is a SET of well-defined instructions to solve a particular problem. It takes a SET of inputs and produces the desired output. FOR example,

An algorithm to subtract two number:

1. Take two number inputs
2. Subtract number using the – operator
3. Display the result

Depending on what you wish to do, it can be straightforward or complex. By using the process of making a novel recipe as an example, it can be understood. To prepare a new dish, one must read the directions and procedures and follow them sequentially, one at a time. The new meal is cooked to perfection as a result of this process.

Step 1: Start

Step 2: Present the greeting screen

Step 3: Add a user preference

Step 4: If the user input is 1, continue to step 5, if it is 2, continue to step 9, if it is 3, continue to step 13, and if it is not, continue to step 3.

Step 5: Present the list of costumes and request the serial number and quantity for rental purposes.

Step 6: Inquire as to whether the user wants to rent any more costumes.

Step 7: If the user desires a more elaborate outfit, proceed to step 5, and if not, go to step 8.

Step 8: Request a username, address, and phone number before generating the rent bill.

Step 9: Present the list of costumes and request the serial number and amount for returns.

Step 10: Inquire as to whether the user wants to return any further costumes.

Step 11: Proceed to step 9 if the user wants to return more costumes, and step 12 if the user doesn't want to return further costumes.

Step 12: In order to generate the return and rent bill, please provide your username, address, and phone number.

Step 13: Show the welcome screen before closing the program.

Step 14: Stop

B. Flowchart

A flowchart is a type of graphically diagram that represents an algorithm, workflow, or process. The flowchart shows that steps as boxes of various kinds, and their codes by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a solution model to a given problem. Flowcharts are used in analysing, designing, Documenting, or managing a process or program in various fields.

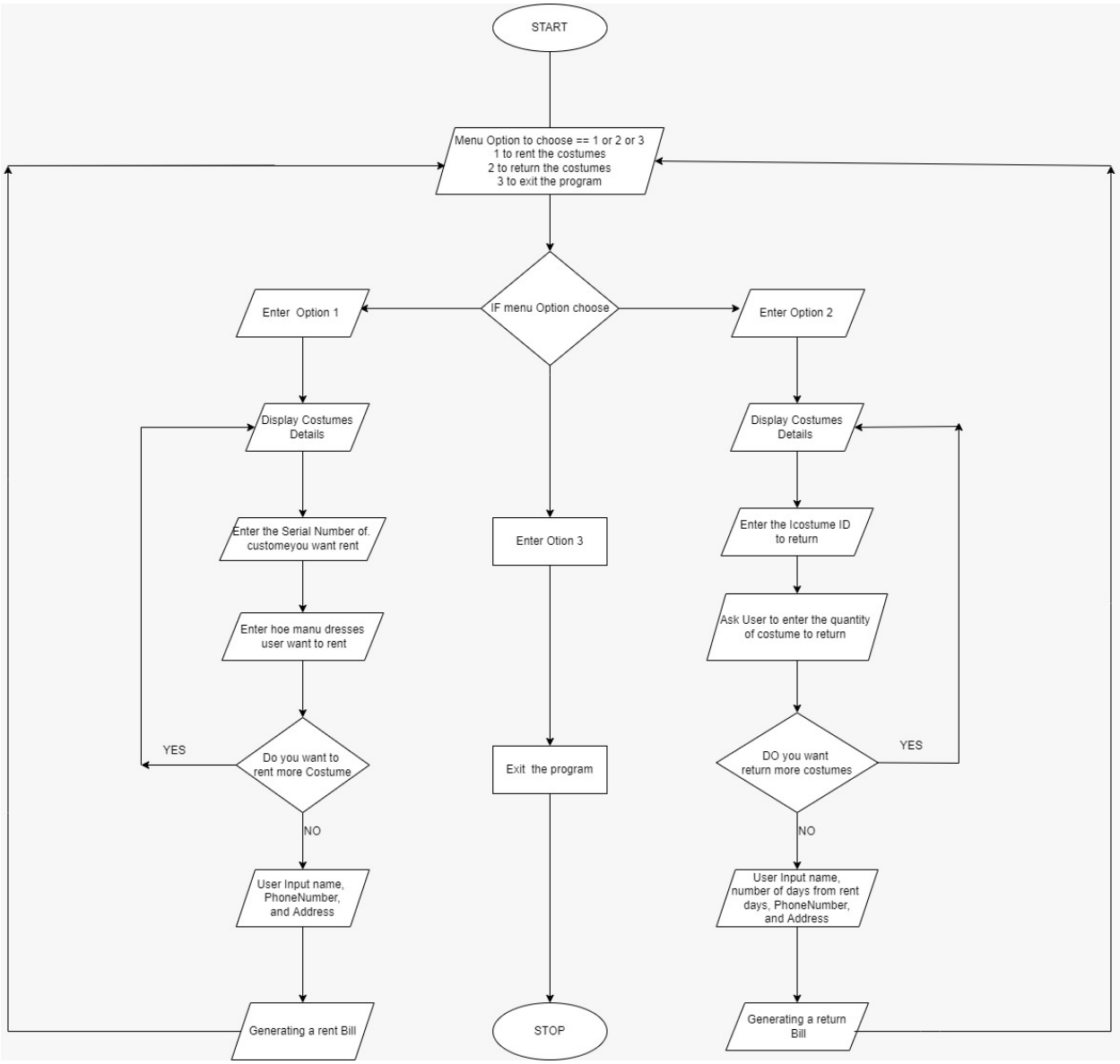

Figure 1: Flowchart of the program

C. Pseudocode

Pseudocode is more like an algorithmic representation of the code involved. It is a way of expressing an algorithm without conforming to specific syntax rules. By learning to read a write pseudocode, you can easily communicate ideas and concepts to other programmers, even though they may be using completely different languages.

1. Main class

START
IMPORT rent, **RETURN**C
CREATE Non parameterized function name as welcome
PRINT

PRINT an empty line
PRINT Welcom**e** to costume rental shop
PRINT an empty line
PRINT ---
CREATE non parameterized function name as display
WHILE True
PRINT Select a desirable option
PRINT (1) || Press 1 to rent a costume.
PRINT (2) || Press 2 to **RETURN** a costume.
PRINT (3) || Press 3 to exit.
CREATE a variable name userOption and **INPUT** Enter a Option
IF userOption is equal to 1

PRINT Let's rent a costume

CALLING a non-parameterized function named as rent.rentCostume()

ELIF selecte**DO**ption is equal to 2
PRINT Let's **RETURN** a costume
CALLING a non-parameterized function named as **RETURN**C.**RETURN**Costume()

ELIF selecte**DO**ption is equal to 3
PRINT Thank You **FOR** Visting Our shop."
EXIT
ELSE
PRINT Invalid input !!
PRINT Please select from the given options
ENDIF
CALLa function welcome
CALL a function display
END

2. Rent class

START
IMPORTdatetime
CREATE a non-parameterized function name as extractingContent
DO
OPEN File
CREATE a content with file.readlines
CLOSE file
RETURN content
CREATE a parameterized function **CREATE**Dictionary by Content
CREATE an empty **SET** of dictionary
FOR index in range upto length of Content)
Dictionary[index+1] is equal to fileContent[index].repalcing \n .spliitng by ,
RETURN dictionary
CREATE a parameterized function name ENby dictionaryData
PRINT S.No., "Costume Name", "\t\t", "Brand", "\t\t\t", "Price", "\t\t", "Quantity"
FOR key,value in dictionaryData.items():
PRINTstr(key)+str("."), "\t", value[0], "\t\t", value[1], "\t\t", value[2], "\t\t", value[3]
RETURN
CREATE a parameterized function name ValidSno by dictionaryData
CREATE a variable validSno with False value
WHILE validSno is equal to False
DO
CREATE a variable SNo and INPUT Enter the Serial Number:

TRY
IF SNo.isdigit()
CREATE a variable SNo as int SNo
IF SNo is grater than 0 and SNo is les than equal length of dictionaryData
IF int mainData[SNo][3] is equal to 0
PRINT an empty line
PRINT out of stock!
PRINT an empty line
PRINT Wanna **TRY** another Costume!
PRINT an empty line
PRINT PRINT Costumes(dictionaryData)

DO
ENDIF
ELSE

validSno = True
PRINT f"The serial number of the costume is {SNo}.
PRINT an empty line

PRINT The costume is available. ")

RETURN SNo
ENDELSE

ELSE
PRINT Please enter a option from the given options only!
ENDELSE

ELSE
PRINT Please type a number !
ENDELSE
EXCEPT
PRINT Invalid serial number
EXCEPTEND
ENDO
ENDWHILE

CREATE a parameterized function name ValidQuantity by dictionaryData, SNo
CREATE an empty list name cart
CREATE an empty list tempRent

CREATE an empty list costumeName
CREATE an empty list costumeBrand
CREATE an empty list costumeNumber
CREATE a variable validQuantity as false
WHILE validQuantity is equal to false
DO
CREATE a variable quantity and INPUT How many dresses you want to rent?
TRY
IF quantity.isdigit
Quantity is equal to int quantity
IF quantity is greater than 0 and quantity is less than equal to int dictionaryData[SNo] [3]
validQuantity is equal to true
dictionaryData [SNo][3] is equal to String (int(dictionaryData [SNo] [3] - quantity))
RETURN quantity
ENDIF
ELIF quantity is graeter than int(dictionaryData[SNo][3]
PRINT Quantity you want is greater than we have in stock.
ELSE
PRINT Invalid Input
ENDELSE
ELSE
PRINT Please enter a number
ENDELSE
ENDIF
EXCEPT
PRINT invalid Quantity !
ENDEXCEPT

CREATE a non parameterized function named as rentcostume
CREATE a variable userWantsClothes as True
CREATE an empty list as Cart
CREATE an empty list as tempRent
CREATE an empty list costumeName
CREATE an empty list costumeBrand
CREATE an empty list costumeNumber

WHILE userWantsClothes is equal to true
DO
PRINT PRINT Costumes with dictionaryData

INITIALIZE SNo as ValidSNowith dictionaryData
INITIALIZE quantity as ValidQuantity with dictionaryData and SNo
CREATE a variable flag as TRUE
FOR costume in cart
IF costume index 0 is equalt to SNo variable
Costume[1] +qquantity

Flag is equal to false
IF flag
APPEND dictionaryData [SNo][0] and quantity to cart
APPEND dictionaryData [SNo][0], dictionaryData [SNo][1], mainData[SNo][2] and quantity to tempRent
APPEND dictionaryData[SNo][0] to costumeName
APPEND dictionaryData[SNo][1] to costumeBrand
APPEND quantity to costumeNumber
Valid_input is equal to false
WHILE valid_input is equal to false
DO
INITIALIZE a variable rentAnother and INPUT wanna rent more(yes/no)?
IF rentAnother is equal to Yes and **CONVERT** to lower

PRINT Youe cart: {cart}
Valid_input is equal to true
BREAK
ENDIF
ELIF rentAnother is equal to no and **CONVERT** to lower
CALL generateBill with tempRent , costumeName, costumeBrand, and costumeNumber
INITALIZE usewantclothes as false
INITALIZE Valid_input as false
ENDE**IF**
ELSE
PRINT Invalid Input !!
DO
ENDELSE
ENDDO
ENDWHILE
CALL updateTextFile with dictionaryData function
ENDDO
ENDWHILE

CREATE a parameterize function named as generatRentBill by tempRent, costumeName, costumeBrand, and costumeNumber

INITIALIZE validName as False
WHILE validName is equal to false
DO
INITIALIZE a variable name as String and INPUT Enter Your Name
IF Name. replace “” isalpha
Validname is equal to True
ENDIF
ELSE
PRINT You entered your name wrong
ENDELSE
INITIALIZE a vartiable address as String and INPUT Enter your address
ENDO
ENDWHILE

ValidPhoneNumber is equal to False
WHILE validPHonenumber is equal to false
DO
INITIALIZE a variable phonenumber as String and INPUT Enter you PhoneNumber
IF phoneNumber. Isdigit
ValidPhoneNumber is equal to true
ENDIF
ELSE
PRINT You entered your phone number wrong
ENDELSE
ENDDO
ENDWHILE

INITIALIZE a variable datatime to datetime.datetime.now
PRINT Rent Bill Details

INITILIZE a variable finalPrice as 0
FOR i in range as length of tempRent

FOR j in range as length as tempRent[i]
INITIALIZE variable **DO**llarpice as Float with tempRentBill[i][2].replace("$","")
INITIALIZE variable priceDetail as **DO**llaprice * tempRentBill[i][3]
INITIALIZE variable finalprice to finalprice +pricedetails

PRINT counter,"\t",row, "\t", priceDetail
Finalprice is equal to finalprice + priceDetail
INITAILIZE a variable row as Empty string
PRINT Name of customer:",Name

PRINT Address:",address
PRINT Number:",phoneNumber
PRINT Date Time of borrow:",dateTime
PRINT Total price is: $"+str(finalPrice)
PRINT Items in rent are:",costumeName
PRINT Brand of Items are:",costumeBrand
PRINT Number of Items in rent are:",costumeNumber
PRINT Bill is also generated in txt file

Text is equal to Rent –{Name} to txt file
OPEN file
SET file.write("\n")
SET file.write Rent Bill Details
SET file.write("\n")
SET file.write f"Name of customer: {Name}
SET filewrite("\n")
SET file.write f"Address: {address}
SET file.write("\n")
SET f"Number: {phoneNumber} }
SET file.write("\n")
SET file.write f"Date Time of borrow: {dateTime}
SET file.write("\n")
SET file.write f f"Total price is: ${finalPrice} }
SET file.write("\n")
SET file.write f"Items in rent are: {costumeName}
SET file.write("\n")
SET file.write f"Brand of Items are: {costumeBrand}
SET file.write("\n")
SET file.write f"Number of Items in rent are: {costumeNumber}

CREATE a parameterize function updateTextFile by dictionaryData
OPEN file
FOR I in mainData. Value
file.write(str(value[0]) + "," + str(value[1]) + "," + str(value[2]) + "," + str(value[3]) + "\n")
CLOSE file
SET Content to extractingContent
SET dictionaryData to **CREATE**Dictionary by parameterContent

3. **RETURN**C class

START
IMPORT datatime
CREATE a non parameterize function name as extractingcontent
OPEN file
SET coonten as file.readlines
CLOSE file
RETURN content
CREATE a parameterize function name as createDictionary by content
SET a variable dictionary as empty set
FOR index in range upto length of Content
INITIALIZE a variable dictionary[index+1] fileContent[index].replaceing “” and spliting by ,
RETURN dictionary
CREATE a parameterize function name as returnCostume by dictionaryData
PRINT S.No.", "\t", "Costume Name", "\t\t", "Brand", "\t\t\t", "Price", "\t\t", "Quantity
FOR key,value in dictionaryData.item
PRINT str(key)+str("."), "\t", value[0], "\t\t", value[1], "\t\t", value[2], "\t\t", value[3]
RETURN
CREATE a parameterize function name as ValidID by dictionaryData
INITIALIZE validID as false
WHILE validID is equal to false
DO
INITIALIZE a variable ID and INPUT Enter the costumeID to **RETURN**
IF ID.digit
INITIALIZE a variable ID as Integer ID
IF ID is greater than 0 and ID is less than equal to length of dictionaryData
INITALIZE validID to True
RETURN ID
BREAK
ENDIF
ELSE
PRINT It appears that you entered an option that was not available. **ENDELSE**
ELSE
PRINT Please type a number next time.
ENDELSE

ENDDO
ENDWHILE

CREATE a parameterize function name as ValidreturnQuantity by dictionaryData and ID

INITAILIZE returnCostumeName as empty list
INITAILIZE returnCostumeBrand as empty list
INITAILIZE returnCostumeNumber as empty list

INITIALIZE validQuantityto false
WHILE validQuantityis equal to False
INITIALIZE a variable quantity and INPUT Enter the quantity you want to **return**:
IF quantity. Isdigit
INITIALIZE quantity as INT quantity
INITIALIZE validQuantity to True
SET dictionaryData [ID][3] as int(dictionaryData [ID][3])
APPEND dictionaryData[ID][0] to returnCostumeName
APPEND dictionaryData[ID][1] to returnCostumeBrand
APPEND qunatity to returnCostumeNumber
RETURN quantity
ENDIF

ELSE
PRINT Please enter a number not anything **ELSE**!
ENDELSE
ENDWHILE

CREATE a non parameterize function name as returnCostume
INITIALIZE variable userretrurnClothes to false
INITIALIZE a variable returnCostumeName as empty list
INITIALIZE a variable returnCostumeBrand as empty list
INITIALIZE a variable returnCostumeNumber as empty list

WHILE userreturnClothes is equal to True
PRINT PRINT Costumes(dictionaryData)
SET variable ID as getValidID with dictionaryData
SET variable quantity as ValidreturnQuantity with dictionaryData and ID
CREATE a variable flag as TRUE
FOR costume in returnCostumeName
IF costume index 0 is equalt to ID variable
Costume[1] +q=quantity

Flag is equal to false
IF flag
APPEND dictionaryData [ID][0]] to returnCostumeName

APPEND dictionaryData [ID][1]] to returnCostumeBrand
APPEND dictionaryData quantity to returnCostumeNumber

Valid_input is equal to false
WHILE valid_input is equal to false
DO
INITIALIZE a variable returnMore and INPUT wanna **RETURN** more(yes/no)?
IF returnMore is equal to Yes and **CONVERT** as lower
Valid_input is equal to true
BREAK
ENDIF
ELIF returnMore is equal to no and **CONVERT** as lower
CALL generatereturnBill with returnCostumeName, returnCostumeBrand, and returnCostumeNumber
INITALIZE userreturnclothes as false
INITALIZE Valid_input as True
ENDE**IF**
ELSE
PRINT Please enter a option from given options only!
DO
ENDELSE
ENDDO
ENDWHILE
CALL updateTextFile with dictionaryData function
ENDDO
ENDWHILE
CREATE a parameterize function name as generatereturnBill by returnCostumeName, returnCostumeBrand, and returnCostumeNumber
INITIALIZE variable validname as false
WHILE validName is equal to false
DO
INITIALIZE a variable Name as String and INPUT enter your name
IF Name.replace(" ", "").isalpha():
INITIALIZE a variable validName toTrue
ENDIF
ELSE
PRINT You entered your name wrong.
ENDELSE

```
    ENDDO
ENDWHILE


INITIALIZE variable  validphonenumber as false
WHILE validphonenumber is equal to false
  DO
      INITIALIZE a variable phonenumber and INPUT enter your phone number
      IF phonenumber.isdigit
          INITIALIZE a variable validphonenumber to True
      ENDIF
      ELSE
          PRINT You entered your phone number wrong.
      ENDELSE
  ENDDO
ENDWHILE
INITIALIZE a variable address as String and INPUT enter your address


INITIALIZE variable  validDay as false
WHILE validDay is equal to false
  DO
      INITIALIZE a variable day and INPUT enter number of Day from rent days:
      IF day.isdigit
          INITIALIZE a variable day to INT day
          INITIALIZE variable  validDay to false
      ENDIF
      ELSE
          PRINT Please enter the days which is always inpositive number!
      ENDELSE
  ENDDO
ENDWHILE


INITIALIZE a variable datatime to datetime.datetime.now

PRINT Rent Bill Details

IF day is greater than 5
      SET fday to day minus 5
      SET fine to fday * 10
ENDIF
```

ELSE
SET fine to 0
ENDELSE

PRINT Name of customer:",Name
PRINT Address:",address
PRINT Number:",phoneNumber
PRINT Date Time of borrow:",dateTime
PRINT Total fine is: $"+str(fine)
PRINT Items in rent are:", returnCostumeName
PRINT Brand of Items are:", returnCostumeBrand
PRINT Number of Items in rent are:", returnCostumeNumber
PRINT Bill is also generated in txt file

Text is equal to Rent –{Name} to txt file
OPEN file
SET file.write("\n")
SET file.write Rent Bill Details
SET file.write("\n")
SET file.write f"Name of customer: {Name}
SET filewrite("\n")
SET file.write f"Address: {address}
SET file.write("\n")
SET f"Number: {phoneNumber} }
SET file.write("\n")
SET file.write f"Date Time of borrow: {dateTime}
SET file.write("\n")
SET file.write f f"Total price is: ${finalPrice} }
SET file.write("\n")
SET file.write f"Items in rent are: { returnCostumeName}
SET file.write("\n")
SET file.write f"Brand of Items are: { returnCostumeBrand}
SET file.write("\n")
SET file.write f"Number of Items in rent are: { returnCostumeNumber}

CREATE a parameterize function updateTextFile by dictionaryData
OPEN file
FOR I in dictionaryData. Value
file.write(str(value[0]) + "," + str(value[1]) + "," + str(value[2]) + "," + str(value[3]) + "\n")
CLOSE file
SET Content to extractingContent

SET dictionaryData to createDictionary parameter Content
ENDDO

D. Data Structure

Data structures are the fundamental constructs around which you build your programs. Each data structure provides a particular way of organizing data so it can be accessed efficiently, depending on your use case. python ships with an extensive SET of data structure in its standard library.

Primitive Date Types

Primitive data types are the most fundamental data structures. They serve as the foundation for data modification. Strings, floating, Booleans, and integers are the four primitive kinds.

i. Integer

Integers are used to represent numerical data, especially whole numbers. Negative whole numbers are also possible. We can use an underscore to divide words in our variable name when there are several of them. Everywhere we utilize the Library Management System program, we must ask for the entire amount as input, even when calculating quantities that are decreasing and increasing. Additionally, integer is utilized when we wish to ask the user for an integer value, such as a menu item in the library. The costume ID is required for both renting and returning costumes.

```
def validSNo(dictionaryData):
    validSNo = False
    while validSNo == False:
        SNo = input("Enter the Serial number: ")
        try:
            if SNo.isdigit():
                SNo = int(SNo)
                if SNo > 0 and SNo <= len(dictionaryData):
                    if int(dictionaryData[SNo][3]) == 0:
                        print("\n")
                        print("Out of Stock! ")
                        print("\n")
                        print("Wanna try another Costume?")
                        print("\n")
                        print(printCostumes(dictionaryData))
                        continue
                    else:
                        validSNo == True
                        print("The serial number of Costume is",SNo)
                        print("\n")
                        print("The Costume is available.")
                        print("\n")
                    return SNo
                else:
                    print("Please enter a option from the given options only!")
                    print("\n")
```

Figure 2: Screenshot of Program showing use of Integer data type

ii. Float

The symbol for floating-point numbers is afloat. It functions with rational numbers like 2.1 and 5.5 as well as decimals. Although it also supports decimal numbers, float is basically identical to integer. As there may be float numbers in the cost of costumes in the shop, a float is utilized in the program to determine the cost to be paid while renting and returning the costumes.

```
finalPrice = 0
for i in range(len(tempRent)):
    for j in range(len(tempRent[i])):
        dollarprice = float(tempRent[i][2].replace("$",""))
        priceDetail = dollarprice * tempRent[i][3]
    finalPrice = finalPrice + priceDetail
```

Figure 3: Screenshot of Program showing use of Float data typ

iii. String

Character data sequences are known as strings. The definition of the string term in Python is str. String literals can be separated using either single or double quotes. Between the beginning delimiter and the corresponding closing delimiter, the entire text is shown. The value of a string can be either a number, an alphabet, or any other symbol, however in this case the value is more like to an alphabet than a number. The number is now devoid of all addition, subtraction, and other features. The names of the brands of the costumes as well as their quantities and prices are written in a string in the application for the costume management system.

```
def updateTextFile(dictionaryData):

    file = open("costumes.txt", "w")
    for value in dictionaryData.values():
        file.write(str(value[0]) + "," + str(value[1]) + "," + str(value[2]) + "," + str(value[3]) + "\n")
    file.close()
```

Figure 4: Screenshot of Program showing use of String data type

iv. Boolean

A Boolean data type has two potential values: True and False. When creating conditional and comparison expressions, they can be helpful. The Boolean type is one of the built-in data types in Python. It serves to illustrate the truth value of an expression. A Boolean data type, which can only take the values true or false, is declared with the bool keyword. When the value is returned, true is equal to 1, and false to 0.

```
valid_input = False
while valid_input == False:
    rentAnother = input("Wanna rent more(yes/no)? ")
    if rentAnother.lower() == "yes":
        print("\n")
        print(f"Your Cart: {cart}")
        print("\n")
        valid_input = True
        break
    elif rentAnother.lower() == "no":
        print("\n")
        generateRentBill(tempRent, costumeName, costumeBrand, costumeNumber)
        userWantsClothes = False
        valid_input = True
    else:
        print("Invalid Input !!")
        print("\n")
        continue
```

Figure 5: Screenshot of Program showing use of Boolean data type

Collection Data Types

We have used a few collection data kinds in this software. List, Dictionary, Set, and Tuple are the four different forms of collection data types. List and Dictionary have been employed as collection data types in this software.

i. List

One of the most popular collection data types is the list. The list's data entries are arranged in chronological order with the appropriate index. Lists are

changeable data types, making it simple to modify them. The application uses a list to display the name of the costume, the brand, the amount of the costume, the cost, and while writing.

```
def validQuantity(dictionaryData, SNo):

    cart = []
    tempRent = []
    costumeName = []
    costumeBrand = []
    costumeNumber = []
    validQuantity = False

    while validQuantity == False:
        quantity = input("How many dresses you want to rent? ")
        try:
            if quantity.isdigit():
                quantity = int(quantity)
                if quantity > 0 and quantity <= int(dictionaryData[SNo][3]):
                    validQuantity = True
                    dictionaryData[SNo][3] = str(int(dictionaryData[SNo][3])- quantity)
                    return quantity
                elif quantity > int(dictionaryData[SNo][3]):
                    print("Quantity you want is greater than we have in stock.")
                else:
                    print("Invalid Input!")
```

Figure 6: Screenshot of Program showing use of List data type

ii. Dictionary

Data values are kept as key:value pairs in dictionaries. A dictionary is a group of items that are unique, changing, and ordered*. Curly brackets are used when writing dictionaries, and they contain keys and values. Items in the dictionary can be changed, are ordered, and cannot be duplicated. Dictionary elements are presented as key:value pairs, and the key name can be used to refer to any of the items.

```
def createDictionary(content):
    dictionary = {}
    for index in range(len(content)):
        dictionary[index+1] = content[index].replace("\n","").split(",")
    return dictionary

def printCostumes(dictionaryData):
    print("-------------------------------------------------------------------------------------")
    print("S.No.", "\t", "Costume Name", "\t\t", "Brand", "\t\t\t", "Price", "\t\t", "Quantity")
    print("-------------------------------------------------------------------------------------")
    for key, value in dictionaryData.items():
        print(key, "\t", value[0], "\t\t", value[1], "\t\t", value[2], "\t\t", value[3])
    print("-------------------------------------------------------------------------------------")

    return ""
```

Figure 7: Screenshot of Program showing use of Dictionary data type

3. Program

A. Implementation of Program

The project, which will help the costume company monitor its inventory, was built using Python programming. This software was developed using a variety of data structures. The project demonstrates how to create a user interface for a costume rental system without the use of any Python GUI toolkits; instead, the interface is easily created, and the output is displayed on the terminal. This program was created by creating numerous modules and functions for a variety of purposes. The three different types of modules in the system regulate how the costume management system operates. The last one is a file named costumes.txt that contains details about the costumes, like their name, brand, number, and cost. They are the Main Module, Rent Module, Return Module, and lastly. The system also makes use of the concept of structures to specify the costume accessories. Every costume that is offered from the text file is printed by the application after it has scanned it. User input determines how the application functions, including whether the user wants to access the main menu, choose to rent a costume, or return the rented costume. The user must select to shut off the program; it is not shut down as part of the costume rental or return procedure. To display the costumes and the quantity of them that are available, the application can read a text file. When a costume is rented or returned, it might also update. Each time a costume is borrowed or returned by a user, a new text file containing the invoice is created. The application uses exception handling to prevent closing if the user enters the incorrect value.

B. Rent, Return and Bill Generate

Rent

This module creates the rent function, which asks users for details like their user name and contact information when they rent costumes. The User Name and Phone Number are also verified. When a user wants to select a costume for rental, the costume ID must be entered. The notification or the invoice are then created using the text file. Each time rent is paid by a different user, a special notification is issued. When a user rents a costume, the rent function is invoked, which reads the text file and adds information about the costume there. Also When a user leases a costume, the number of existing costumes in the file is decreased by the number of costumes the user has rented. The authentication for renting additional costumes and costume ID is also finished in order to avoid this and guarantee the program operates smoothly. The numerous costume rent condition is also implemented using Python loops and the conditional function of the rent module. In order to avoid confusion, if someone decides to rent many costumes, they should put each one on the note along with the total amount, which is also calculated on the note.

```
Let's rent a costume


-------------------------------------------------------------------------------
S.No.    Costume Name            Brand                   Price           Quantity
-------------------------------------------------------------------------------
1        Iron Man                MegaPlex                $14.5           200
2        Thor Costume            DollarSmart             $18             100
3        Hulk Costume            Marvel INC              $23             100
4        Capt America            DC Comics               $25             200
-------------------------------------------------------------------------------

Enter the Serial number: 1
The serial number of Costume is 1


The Costume is available.


How many dresses you want to rent? 20
```

Figure 8: Screenshot of program showing renting process

```
========================================
            Rent Bill Details
========================================
Name of customer: Anju Yadav
Address: Kamalpokhari
Number: 9852346795
Date Time of borrow: 2022-08-26 13:27:06.761632
Total price is: $340.0
Items in rent are: [['Iron Man'], ['Capt America']]
Brand of Items are: [['MegaPlex'], ['DC Comics']]
Number of Items in rent are: [[20], [2]]
----------------------------------
Bill is also generated in txt file.
----------------------------------
```

Figure 9: Screenshot of program printing Rent Bill in terminal

```
========================================
            Rent Bill Details
========================================
Name of customer: Anju Yadav
Address: Kamalpokhari
Number: 9852346795
Date Time of borrow: 2022-08-26 13:27:06.761632
Total price is: $340.0
Items in rent are: [['Iron Man'], ['Capt America']]
Brand of Items are: [['MegaPlex'], ['DC Comics']]
Number of Items in rent are: [[20], [2]]
```

Figure 10: Screenshot of program printing Rent Bill in txt format

Return

This module creates the return method, which asks for details such a user name and phone number when a user returns outfits. The User Name and Phone Number are also verified. When a user wants to select a costume for return, the costume ID must be entered. When a user returns a costume, the quantity of costumes in the file is updated by adding the number of returned costumes to the quantity of costumes already there. The borrower's name and costume ID are requested when the user returns the outfits. The outfit can only be borrowed for a maximum of 5 days, and if the return date has passed, a fine of $10 is applied every day, which is presented with a message after totalling the total.

```
Let's return a costume


-------------------------------------------------------------------------------
S.No.    Costume Name           Brand                  Price          Quantity
-------------------------------------------------------------------------------
1        Iron Man               MegaPlex               $14.5          180
2        Thor Costume           DollarSmart            $18            100
3        Hulk Costume           Marvel INC             $23            100
4        Capt America           DC Comics              $25            198
-------------------------------------------------------------------------------

Enter the costume ID to return: 1
Enter the quantity you wanna return: 20
```

Figure 11: Screenshot of program showing returning process

```
==========================================
            Return Bill Details
==========================================
Name of customer: Anju Yadav
Address: Kamalpokhari
Number: 9852465795
Date Time of return: 2022-08-26 13:36:15.385009
Total fine: $200
Items in rent are: [['Iron Man']]
Brand of Items are: [['MegaPlex']]
Number of Items in rent are: [[20]]
----------------------------------
Bill is also generated in txt file.
----------------------------------
```

Figure 12: Screenshot of program printing Return Bill in terminal

```
==========================================
            Return Bill Details
==========================================
Name of customer: Anju Yadav
Address: Kamalpokhari
Number: 9852465795
Date Time of return: 2022-08-26 13:36:15.385009
Total fine: $200
Items in rent are: [['Iron Man']]
Brand of Items are: [['MegaPlex']]
Number of Items in rent are: [[20]]
```

Figure 13: Screenshot of program printing Return Bill in txt format

4. Testing

a. Test 1

Implementation of TRY and EXCEPT

Test No.	1
Objective	To test implementation of **TRY-EXCEPT**
Action	➢ Costume Menu Press 1 to rent a costume. Press 2 to return a costume. Press 3 to exit. ➢ Enter a option: 9 When the user gives 9 as input in Costume Menu
Expected Result	The message for invalid input should be shown
Actual Result	The message for invalid input was shown.
Conclusion	The test was Successful.

Table 1: Test table of implementation of try-except

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

            Welcome to Costume Rental Shop

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


Select a desirable option
(1) || Press 1 to rent a costume.
(2) || Press 2 to return a costume.
(3) || Press 3 to exit.


Enter a option: 9


Invalid input !!
Please select from the given Options.
```

Figure 14: screenshot of implementation of try-except

b. Test 2

Selection of renting and RETURNing of customes:

Test No.	2(i)
Objective	To test rent with negative and non-existent input number
Action	1. Enter the Serial number: -1 2. How many dresses you want to rent? 0
Expected Result	The message for invalid input should be shown.
Actual Result	The message for invalid input was shown.
Conclusion	The test was Successful.

Table 2: Test table of rent and return process

```
Let's rent a costume


-------------------------------------------------------------------------------
S.No.    Costume Name           Brand                  Price          Quantity
-------------------------------------------------------------------------------
1        Iron Man               MegaPlex               $14.5          200
2        Thor Costume           DollarSmart            $18            100
3        Hulk Costume           Marvel INC             $23            100
4        Capt America           DC Comics              $25            198
-------------------------------------------------------------------------------

Enter the Serial number: -1
Please type a number!
```

Figure 15: Screenshot of program while providing negative value as input

```
Let's rent a costume


-------------------------------------------------------------------------------
S.No.    Costume Name           Brand                  Price          Quantity
-------------------------------------------------------------------------------
1        Iron Man               MegaPlex               $14.5          200
2        Thor Costume           DollarSmart            $18            100
3        Hulk Costume           Marvel INC             $23            100
4        Capt America           DC Comics              $25            198
-------------------------------------------------------------------------------

Enter the Serial number: -1
Please type a number!


Enter the Serial number: 1
The serial number of Costume is 1


The Costume is available.


How many dresses you want to rent? 0
Invalid Input!
```

Figure 16: Screenshot of program while providing non existed value as input

c. Test 3

File generation of renting costumes:

Test No.	3
Objective	To show file generation of rent
Action	1. Enter the Serial Number: 1 2. How many dresses you want to rent? 50
Expected Result	The rent-user.txt file will be generated.
Actual Result	The rent-user.txt file is generated.
Conclusion	The test was Successful.

Table 3: Test table of file generation of rent process

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

            Welcome to Costume Rental Shop

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


Select a desirable option
(1) || Press 1 to rent a costume.
(2) || Press 2 to return a costume.
(3) || Press 3 to exit.


Enter a option: 1


Let's rent a costume


-------------------------------------------------------------------------------------
S.No.    Costume Name           Brand                  Price          Quantity
-------------------------------------------------------------------------------------
1        Iron Man               MegaPlex               $14.5          200
2        Thor Costume           DollarSmart            $18            100
3        Hulk Costume           Marvel INC             $23            100
4        Capt America           DC Comics              $25            198
-------------------------------------------------------------------------------------

Enter the Serial number: 1
The serial number of Costume is 1


The Costume is available.


How many dresses you want to rent? 50
```

Figure 17: Screenshot of program while renting the costume

```
==========================================
             Rent Bill Details
==========================================
Name of customer: Archana Yadav
Address: Siraha
Number: 9874563210
Date Time of borrow: 2022-08-26 14:11:33.020915
Total price is: $725.0
Items in rent are: [['Iron Man']]
Brand of Items are: [['MegaPlex']]
Number of Items in rent are: [[50]]
----------------------------------
Bill is also generated in txt file.
----------------------------------
```

Figure 18: Screenshot of Bill of rent in the terminal

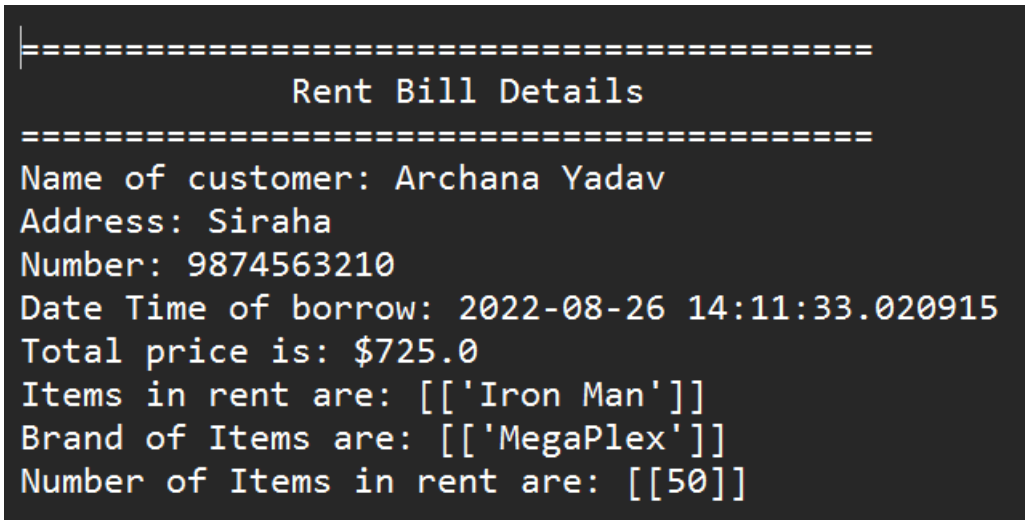

Figure 19: Screenshot of Bill of rent in the txt form

d. Test 4

File generation of Returning process of customs.

Test No.	4
Objective	To show file generation of return
Action	1. Enter the costume ID to return: 1 2. Enter the quantity you wanna return: 50
Expected Result	The return-user.txt file will be generated.
Actual Result	The return-user.txt file is generated.
Conclusion	The test was successful.

Table 4: Test table of file generation of return process

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

            Welcome to Costume Rental Shop

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


Select a desirable option
(1) || Press 1 to rent a costume.
(2) || Press 2 to return a costume.
(3) || Press 3 to exit.


Enter a option: 2


Let's return a costume


-------------------------------------------------------------------------------
S.No.    Costume Name           Brand                  Price          Quantity
-------------------------------------------------------------------------------
1        Iron Man               MegaPlex               $14.5          150
2        Thor Costume           DollarSmart            $18            100
3        Hulk Costume           Marvel INC             $23            100
4        Capt America           DC Comics              $25            198
-------------------------------------------------------------------------------

Enter the costume ID to return: 1
Enter the quantity you wanna return: 50
Wanna return more(yes/no)? no
```

Figure 20: Screenshot of file generation of return

```
==========================================
            Return Bill Details
==========================================
Name of customer: Archana Yadav
Address: Siraha
Number: 9856423157
Date Time of return: 2022-08-26 14:19:21.634956
Total fine: $200
Items in rent are: [['Iron Man']]
Brand of Items are: [['MegaPlex']]
Number of Items in rent are: [[50]]
----------------------------------
Bill is also generated in txt file.
----------------------------------
```

Figure 21: Screenshot of Bill of return in the terminal

```
========================================
            Return Bill Details
========================================
Name of customer: Archana Yadav
Address: Siraha
Number: 9856423157
Date Time of return: 2022-08-26 14:19:21.634956
Total fine: $200
Items in rent are: [['Iron Man']]
Brand of Items are: [['MegaPlex']]
Number of Items in rent are: [[50]]
```

Figure 22: Screenshot of Bill of return in the txt form

e. Test 5

The update in stock of customs.

Test No.	5
Objective	Show the update in costume file 1. Show the quantity being deducted while renting the costumes 2. Show the quantity being added while returning the costumes
Action	1. Enter the Serial number: 2 How many dresses you want to rent? 40 2. Enter the costume ID to return: 2 Enter the quantity you wanna return: 40
Expected Result	The costumes file will be updated.
Actual Result	The costumes file is updated.
Conclusion	The test was successful.

Table 5: Test table showing updates in quantity of costumes in txt file

```
Iron Man,MegaPlex,$14.5,200
Thor Costume,DollarSmart,$18,100
Hulk Costume,Marvel INC,$23,100
Capt America,DC Comics,$25,198
```

Figure 23: Screenshot of text file before renting

```
Let's rent a costume


-------------------------------------------------------------------------------
S.No.    Costume Name           Brand                  Price          Quantity
-------------------------------------------------------------------------------
1        Iron Man               MegaPlex               $14.5          200
2        Thor Costume           DollarSmart            $18            100
3        Hulk Costume           Marvel INC             $23            100
4        Capt America           DC Comics              $25            198
-------------------------------------------------------------------------------

Enter the Serial number: 2
The serial number of Costume is 2


The Costume is available.


How many dresses you want to rent? 40
```

Figure 24: Screenshot of program while renting the costume

```
Iron Man,MegaPlex,$14.5,200
Thor Costume,DollarSmart,$18,60
Hulk Costume,Marvel INC,$23,100
Capt America,DC Comics,$25,198
```

Figure 25: Screenshot of txt file after renting

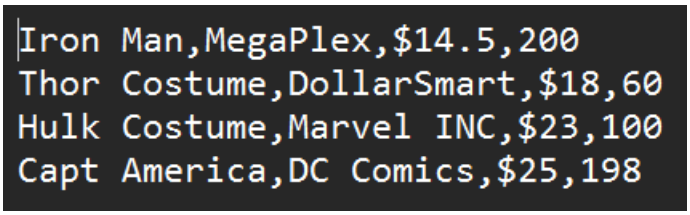

Figure 26: Screenshot of text file before returning

```
Enter a option: 2

Let's return a costume

-------------------------------------------------------------------------------
S.No.    Costume Name            Brand                   Price          Quantity
-------------------------------------------------------------------------------
1        Iron Man                MegaPlex                $14.5          200
2        Thor Costume            DollarSmart             $18            60
3        Hulk Costume            Marvel INC              $23            100
4        Capt America            DC Comics               $25            198
-------------------------------------------------------------------------------

Enter the costume ID to return: 2
Enter the quantity you wanna return: 40
```

Figure 27: Screenshot of program while returning the costume

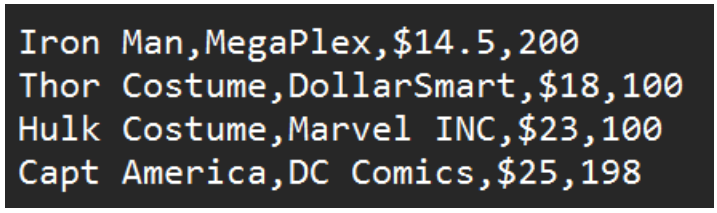

Figure 28: Screenshot of txt file after returning the costume

5. Conclusion

In the second semester, this was the first piece of coursework that was given to us. We have essentially mastered all of Python's fundamentals. After completing this coursework, we learnt a variety of new things, including how to create several projects with Python-like ease. One of these projects was already created throughout our coursework. In a similar vein, it has assisted me in expanding my imagination. Before, I had no idea how to use Python to create the system GUI in the console, but now that I do, we may start to think at that level and learn more about Python's vast capabilities. We can finish this homework after much effort and investigation, which even helped me improve my research abilities. Even more tests were conducted to identify mistakes and bugs in the developed code because it is inevitable that there will be bugs in the program. After much research and effort, and with the aid of the module leader, who continually encouraged me to finish my project, I was able to submit this coursework on time.

I gained new knowledge about the python programming language as well as new information on topics unrelated to python but may be useful in other programming languages, such as how to create flowcharts and pseudocode, from the total coursework. Algorithm and numerous others that could be useful in creating reports and other programming languages in the future. It took me a little longer than I had anticipated to do my assignment, which surprised me. I would want to express my gratitude to the module leaders and the several lecturers that assisted me in finishing the coursework.

Thus, completing this coursework was a fantastic experience that helped us grow as creative people and demonstrated the potential and breadth of the Python programming language. It has aided me in acquiring a variety of abilities and skills that may be useful in the future.

6. Appendix

Main Module

```
import rent
import returnC
def welcome():
    print("------------------------------------------------------------")
    print()
    print("            Welcome to Costume Rental Shop                ")
    print()
    print("------------------------------------------------------------")

def display():

    while True:
        print("\n")
        print("Select a desirable option")
        print("(1) || Press 1 to rent a costume.")
        print("(2) || Press 2 to return a costume.")
        print("(3) || Press 3 to exit.")
        print("\n")
        userOption = input("Enter a option: ")
        if userOption == "1":
            print("\n")
            print("Let's rent a costume")
            print("\n")
            rent.rentCostume()

        elif userOption == "2":
            print("\n")
            print("Let's return a costume")
            print("\n")
            returnC.returnCostume()
```

```
    elif userOption == "3":
        print("\n")
        print("           Thank You for Visiting Our Shop. ")


    else:
        print("\n")
        print("Invalid input !!")
        print("Please select from the given Options.")


welcome()
display()
```

Rent Module

```
import datetime

def extractingContent():
    file = open("costumes.txt","r")
    content = file.readlines()
    file.close()
    return content


def createDictionary(content):
    dictionary = {}
    for index in range(len(content)):
        dictionary[index+1] = content[index].replace("\n","").split(",")
    return dictionary

def printCostumes(dictionaryData):
    print("-------------------------------------------------------------------------------")
    print("S.No.", "\t", "Costume Name", "\t\t", "Brand", "\t\t\t", "Price", "\t\t", "Quantity")
    print("-------------------------------------------------------------------------------")
    for key, value in dictionaryData.items():
        print(key, "\t", value[0], "\t\t", value[1], "\t\t", value[2], "\t\t", value[3])
    print("-------------------------------------------------------------------------------")

    return ""

def validSNo(dictionaryData):
    validSNo = False
    while validSNo == False:
        SNo = input("Enter the Serial number: ")
```

```
        try:
            if SNo.isdigit():
                SNo = int(SNo)
                if SNo > 0 and SNo <= len(dictionaryData):
                    if int(dictionaryData[SNo][3]) == 0:
                        print("\n")
                        print("Out of Stock! ")
                        print("\n")
                        print("Wanna try another Costume?")
                        print("\n")
                        print(printCostumes(dictionaryData))
                        continue
                    else:
                        validSNo == True
                        print("The serial number of Costume is",SNo)
                        print("\n")
                        print("The Costume is available.")
                        print("\n")
                    return SNo
                else:
                    print("Please enter a option from the given options only!")
                    print("\n")
            else:
                print("Please type a number!")
                print("\n")
        except:
            Print("Invalid Serial Number")


def validQuantity(dictionaryData, SNo):

    cart = []
    tempRent = []
    costumeName = []
    costumeBrand = []
    costumeNumber = []
    validQuantity = False

    while validQuantity == False:
        quantity = input("How many dresses you want to rent? ")
        try:
            if quantity.isdigit():
                quantity = int(quantity)
                if quantity > 0 and quantity <= int(dictionaryData[SNo][3]):
                    validQuantity = True
                    dictionaryData[SNo][3] = str(int(dictionaryData[SNo][3])- quantity)
```

```
                return quantity
            elif quantity > int(dictionaryData[SNo][3]):
                print("Quantity you want is greater than we have in stock.")
            else:
                print("Invalid Input!")
        else:
            print("Please enter a number.")
    except:
        print("Invalid Quantity !")


def rentCostume():

    userWantsClothes = True
    cart = []
    tempRent = []
    costumeName = []
    costumeBrand = []
    costumeNumber = []

    while userWantsClothes == True:
        print(printCostumes(dictionaryData))
        SNo = validSNo(dictionaryData)
        quantity = validQuantity(dictionaryData, SNo)
        #print(quantity)

        flag = True
        for costume in cart:
            if costume[0] == SNo:
                costume[1] += quantity
                flag = False
        if flag:
            cart.append([dictionaryData[SNo][0], quantity])
            tempRent.append([dictionaryData[SNo][0],                dictionaryData[SNo][1],
dictionaryData[SNo][2], quantity])
            costumeName.append([dictionaryData[SNo][0]])
            costumeBrand.append([dictionaryData[SNo][1]])
            costumeNumber.append([quantity])

        valid_input = False
        while valid_input == False:
            rentAnother = input("Wanna rent more(yes/no)? ")
            if rentAnother.lower() == "yes":
                print("\n")
                print(f"Your Cart: {cart}")
                print("\n")
```

```
            valid_input = True
            break
        elif rentAnother.lower() == "no":
            print("\n")
            generateRentBill(tempRent, costumeName, costumeBrand, costumeNumber)
            userWantsClothes = False
            valid_input = True
        else:
            print("Invalid Input !!")
            print("\n")
            continue
    updateTextFile(dictionaryData)
    print("\n")

def generateRentBill(tempRent, costumeName, costumeBrand, costumeNumber):

    validName = False
    while validName == False:
        Name = str(input("Enter your name: "))
        if Name.replace(" ", "").isalpha():
            validName = True
        else:
            print("You entered your name wrong.")
            print("\n")

    address = str(input("Enter your Address: "))

    validPhoneNumber = False
    while validPhoneNumber == False:
        phoneNumber = str(input("Enter your Phone Number: "))
        if phoneNumber.isdigit():
            validPhoneNumber = True
        else:
            print("You entered your phone number wrong.")
            print("\n")

    dateTime = datetime.datetime.now()
    print("\n")
    print("========================================")
    print("           Rent Bill Details          ")
    print("========================================")

    finalPrice = 0
    for i in range(len(tempRent)):
        for j in range(len(tempRent[i])):
            dollarprice = float(tempRent[i][2].replace("$",""))
```

```
            priceDetail = dollarprice * tempRent[i][3]
        finalPrice = finalPrice + priceDetail

    print("Name of customer:",Name)
    print("Address:",address)
    print("Number:",phoneNumber)
    print("Date Time of borrow:",dateTime)
    print("Total price is: $"+str(finalPrice))
    print("Items in rent are:",costumeName)
    print("Brand of Items are:",costumeBrand)
    print("Number of Items in rent are:",costumeNumber)
    print("----------------------------------")
    print("Bill is also generated in txt file.")
    print("----------------------------------")

    text = "Rent-"+Name+".txt"
    file = open(text,"w")
    file.write("========================================")
    file.write("\n")
    file.write("            Rent Bill Details          ")
    file.write("\n")
    file.write("========================================")
    file.write("\n")
    file.write(f"Name of customer: {Name}")
    file.write("\n")
    file.write(f"Address: {address}")
    file.write("\n")
    file.write(f"Number: {phoneNumber}")
    file.write("\n")
    file.write(f"Date Time of borrow: {dateTime}")
    file.write("\n")
    file.write(f"Total price is: ${finalPrice}")
    file.write("\n")
    file.write(f"Items in rent are: {costumeName}")
    file.write("\n")
    file.write(f"Brand of Items are: {costumeBrand}")
    file.write("\n")
    file.write(f"Number of Items in rent are: {costumeNumber}")

def updateTextFile(dictionaryData):

    file = open("costumes.txt", "w")
    for value in dictionaryData.values():
        file.write(str(value[0]) + "," + str(value[1]) + "," + str(value[2]) + "," + str(value[3]) +
"\n")
    file.close()
```

```
content = extractingContent()
dictionaryData = createDictionary(content)
```

Return Module

```
import datetime

def extractingContent():
    file = open("costumes.txt","r")
    content = file.readlines()
    file.close()
    return content


def createDictionary(content):
    dictionary = {}
    for index in range(len(content)):
        dictionary[index+1] = content[index].replace("\n","").split(",")
    return dictionary

def printCostumes(dictionaryData):
    print("--------------------------------------------------------------------------------")
    print("S.No.", "\t", "Costume Name", "\t\t", "Brand", "\t\t\t", "Price", "\t\t", "Quantity")
    print("--------------------------------------------------------------------------------")
    for key, value in dictionaryData.items():
        print(key, "\t", value[0], "\t\t", value[1], "\t\t", value[2], "\t\t", value[3])
    print("--------------------------------------------------------------------------------")

    return ""


def validID(dictionaryData):

    validId = False
    while validId == False:
        ID = input("Enter the costume ID to return: ")
        if ID.isdigit():
            ID = int(ID)
```

```
            if ID > 0 and ID <= len(dictionaryData):
                validID = True
                return ID
                break
            else:
                print("It appears that you entered an option that was not available.")
                print("\n")
        else:
            print("Please type a number next time.")
            print("\n")


def validReturnQuantity(dictionaryData,ID):

    returnCostumeName = []
    returnCostumeBrand = []
    returnCostumeNumber = []

    validQuantity = False
    while validQuantity == False:
        quantity = input("Enter the quantity you wanna return: ")
        if quantity.isdigit():
            quantity = int(quantity)
            validQuantity = True
            dictionaryData[ID][3] = str(int(dictionaryData[ID][3]) + quantity)
            returnCostumeName.append([dictionaryData[ID][0]])
            returnCostumeBrand.append([dictionaryData[ID][1]])
            returnCostumeNumber.append([quantity])
            return quantity
        else:
            print("Please enter a number not anything else!")
            print("\n")

def returnCostume():

    userReturnsClothes = True
    returnCostumeName = []
    returnCostumeBrand = []
    returnCostumeNumber = []

    while userReturnsClothes == True:
        print(printCostumes(dictionaryData))
        ID = validID(dictionaryData)
        quantity = validReturnQuantity(dictionaryData,ID)

        flag = True
```

```
        for costume in returnCostumeName:
            if costume[0] == ID:
                costume[1] += quantity
                flag = False
        if flag:
            returnCostumeName.append([dictionaryData[ID][0]])
            returnCostumeBrand.append([dictionaryData[ID][1]])
            returnCostumeNumber.append([quantity])

        valid_input = False
        while valid_input == False:
            returnMore = input("Wanna return more(yes/no)? ")
            if returnMore.lower() == "yes":
                print("\n")
                valid_input = True
                break
            elif returnMore.lower() == "no":
                print("\n")
                generateReturnBill(returnCostumeName,                returnCostumeBrand,
returnCostumeNumber)
                userReturnsClothes = False
                valid_input = True
            else:
                print("Please enter a option from given options only!")
                print("\n")
                continue
        updateTextFile(dictionaryData)
        print("\n")

def              generateReturnBill(returnCostumeName,              returnCostumeBrand,
returnCostumeNumber):

    validName = False
    while validName == False:
        Name = str(input("Enter your name: "))
        if Name.replace(" ", "").isalpha():
            validName = True
        else:
            print("You entered your name wrong.")
            print("\n")


    validPhoneNumber = False
    while validPhoneNumber == False:
        phoneNumber = str(input("Enter your Phone Number: "))
        if phoneNumber.isdigit():
```

```
            validPhoneNumber = True
        else:
            print("You entered your phone number wrong.")
            print("\n")

    address = str(input("Enter your Address: "))

    validDay = False
    while validDay == False:
        day = input("Enter number of Day from rent days: ")
        if day.isdigit():
            day = int(day)
            validDay = True
        else:
            print("Please enter the days which is always inpositive number!")
            print("\n")


    dateTime = datetime.datetime.now()
    print("\n")
    print("========================================")
    print("            Return Bill Details          ")
    print("========================================")

    if day > 5:
        fday = day - 5
        fine = fday*10
    else:
        fine = 0

    print("Name of customer:",Name)
    print("Address:",address)
    print("Number:",phoneNumber)
    print("Date Time of return:",dateTime)
    print("Total fine: $"+str(fine))
    print("Items in rent are:",returnCostumeName)
    print("Brand of Items are:",returnCostumeBrand)
    print("Number of Items in rent are:",returnCostumeNumber)
    print("---------------------------------")
    print("Bill is also generated in txt file.")
    print("---------------------------------")


    text = "Return-"+Name+".txt"
    file = open(text,"w")
    file.write("========================================")
```

```
    file.write("\n")
    file.write("            Return Bill Details          ")
    file.write("\n")
    file.write("=======================================")
    file.write("\n")
    file.write(f"Name of customer: {Name}")
    file.write("\n")
    file.write(f"Address: {address}")
    file.write("\n")
    file.write(f"Number: {phoneNumber}")
    file.write("\n")
    file.write(f"Date Time of return: {dateTime}")
    file.write("\n")
    file.write(f"Total fine: ${fine}")
    file.write("\n")
    file.write(f"Items in rent are: {returnCostumeName}")
    file.write("\n")
    file.write(f"Brand of Items are: {returnCostumeBrand}")
    file.write("\n")
    file.write(f"Number of Items in rent are: {returnCostumeNumber}")

def updateTextFile(dictionaryData):

    file = open("costumes.txt", "w")
    for value in dictionaryData.values():
        file.write(str(value[0]) + "," + str(value[1]) + "," + str(value[2]) + "," + str(value[3]) +
"\n")
    file.close()

content = extractingContent()
dictionaryData = createDictionary(content)
```