



# Module Code & Module Title CS4051NP Fundmental Of Computing

**Assessment weightage & Type 60% Individual Coursework**

**Year and Semester 2023 Spring**

**Student’s Name: Bibek Bot**

**Group: C6**

**College ID: NP04CP4S230019 London Met ID: 22085890 Assessment Due Date: 25th August 2023**

**Assessment Submission Date: 25th August 2023**

# Table of Contents

1. **Introduction ............................................................................................................... 1**

**1.2. Visual Studio Code ............................................................................................. 1**

1. **Discussion and Analysis .......................................................................................... 3** 
   1. **Algorithm ............................................................................................................. 3**
   2. **Flowchart ............................................................................................................. 5 2.3. PseudoCode ........................................................................................................ 6**

**PseudoCode of Display.py ...........................................................................................................6**

**PseudoCode of Write.py .............................................................................................................6**

**PseudoCode of Main.py ..............................................................................................................7**

**PseudoCode of Read.py ..............................................................................................................8**

**PseudoCode of Operation.py ......................................................................................................9**

**2.4. Data Structure ................................................................................................... 12**

1. **Program ................................................................................................................... 15**
2. **Testing ..................................................................................................................... 18**

**Test 1: ....................................................................................................................... 18**

**Test 2 ........................................................................................................................ 19**

**Test of Rent: ............................................................................................................................. 19**

**Test of Return ........................................................................................................................... 20**

**................................................................................................................................................ 21**

**Test 3 ........................................................................................................................ 22**

**Test 4 ........................................................................................................................ 23**

**Test 5 ........................................................................................................................ 25**

1. **Conclusion ............................................................................................................... 27**
2. **Bibliography ............................................................................................................ 28**
3. **Appendix .................................................................................................................. 29**

**Display.py ................................................................................................................. 29**

**Read.py ..................................................................................................................... 29**

**Write.py .................................................................................................................... 30**

**Operation.py ............................................................................................................ 30**

# Table of Figures

**Figure 1: Flowchart ....................................................................................................... 5**

**Figure 2: Integer Data Type ........................................................................................ 12**

**Figure 3: Float Data Type ........................................................................................... 13**

**Figure 4: String Data Type .......................................................................................... 13**

**Figure 5: List Data type ............................................................................................... 14**

**Figure 6: Print Function .............................................................................................. 15**

**Figure 7: Read Function ............................................................................................. 15**

**Figure 8: Append Function ......................................................................................... 16**

**Figure 9: Write function .............................................................................................. 16**

**Figure 10: Input Function ........................................................................................... 16**

**Figure 11: Int Function ................................................................................................ 17**

**Figure 12: Float Function ........................................................................................... 17**

**Figure 13: Print Function ............................................................................................ 17**

**Figure 14: Figure of Test 1: ........................................................................................ 18**

**Figure 15: Enter negative value ................................................................................. 19**

**Figure 16: Entering non existing value ..................................................................... 20**

**Figure 17: Entering negative value in return items .................................................. 20**

**Figure 18: Entering non existing value on return item ............................................ 21**

**Figure 19: Rent bill shown on shell ........................................................................... 22**

**Figure 20: Rented Bill ................................................................................................. 23**

**Figure 21: Return bill shown on shell ........................................................................ 24**

**Figure 22: Returned Bill .............................................................................................. 25**

**Figure 23: After renting equipment............................................................................ 26**

**Figure 24: Updated txt after renting item .................................................................. 26**

**Figure 25: After returning items ................................................................................. 26** **Figure 26: After returning item txt updated file ........................................................ 26**

**Table of Tables**

**Table 1: Test 1 table .................................................................................................... 18**

**Table 2:Test table of rent item ................................................................................... 19**

**Table 3: Test table of return item ............................................................................... 20**

**Table 4: Test 3 table of rent bill .................................................................................. 22**

**Table 5: Test 4 table .................................................................................................... 23**

**Table 6: Table of test 5 ................................................................................................ 25**

## 1. Introduction

This course work is about the equipment renting system. It is made with the flowchart, algorithm, pseudo code, and data structure.

Technology is evolving at a much faster rate than in the past, so we are updating our equipment rental system to keep up. As we all know, in the past, people would collect the data of rented and returned equipment in the bill collected copy. And to see the data of a specific person, we must search all of the collected bill copies to see whether or not he had rented or returned equipment within the time limit, and whether or not he/she had paid the fine for the respected equipment. Such records were difficult to find in the collected bill copies.

As in this time and period there are many advance technology are bring introduce. We have to go along with it as environment and situation are demanding it. As for the demanding equipment management system is made. This equipment management system provides the information about the equipment which are available and about the person who rented and returned the equipment's. Not only has that it also provided the information about the equipment whether or not the rented equipment's are returned in the given period of time if not the customer should pay the fine for the delay day.

The main focus of this course work is to find students' understanding and capability, in this course work students are to make equipment renting system where a person he/she is to rent or return the equipment and weather the rented equipment's are returned in the given period of time if he/she doesn't then they have to pay the fine for the returned equipment's delay. The following technologies are used to construct the renting system.

### 1.2. Visual Studio Code

VS Code is another name for the IDE Visual Studio Code. Visual Studio Code's main functionality, a lightning-fast source code editor, makes it perfect for regular use. With its extensive language support and tools like syntax highlighting, bracket matching, autoindentation, box selection, snippets, and more, Visual Studio Code makes it simple for you to get up and running quickly. With the use of minor modifications, smart keyboard shortcuts, and just plain blocks of text, you can navigate your code with ease. I used this IDE to create the Python code for the community-contributed keyboard shortcut mappings for my coursework. Tools with deeper code understanding are needed for serious coding (Visual Studio Code , 2023).

## 2. Discussion and Analysis

### 2.1. Algorithm

An algorithm is a method for running programmes step by step and in a certain order to produce the desired results. It develops simple language that makes programmes and their execution procedures simpler to comprehend (UpGrad, 2020).

The algorithm of this program is:-

Step 1: Start

Step 2: Choose a given number from 1 to 4

Step 3: If choose 1 display items

Step 4: If choose 2 display items and goto renting process i.e step 4.1

Step 4.1: Enter customer name, customer address, customer contact, serial number, quantity, duration

Step 4.2: Initialize varible item and call find\_item\_by\_serial\_number function and pass item\_list and serial\_number

Step 4.3: If item is none print message item not found

Step 4.4: If quantity is equal to user input quantity print message out of stock

Step 4.5: Calculate grand total with vat

Step 4.6: Store customer name, customer address, customer contact, serial number, quantity, duration, grand totol in invoice

Step 4.7: Updated the quantity

Step 4.8: Write invoice

Step 4.9: Display rented successful

Step 4.10: Print Invoice

Step 5: if choose 3 goto returning process i.e step 5.1

Step 5.1: Enter customer name, serial number, quantity

Step 5.2: Initialize varible item and call find\_item\_by\_serial\_number function and

Pass the item\_list and serial number

Step 5.3: If item is none print message item not found

Step 5.4: Enter rental duration

Step 5.5: Calculate total amount

Step 5.7: Updated the quantity

Step 5.8: Write invoice

Step 5.9: Display return successful

Step 5.10: Print Invoice

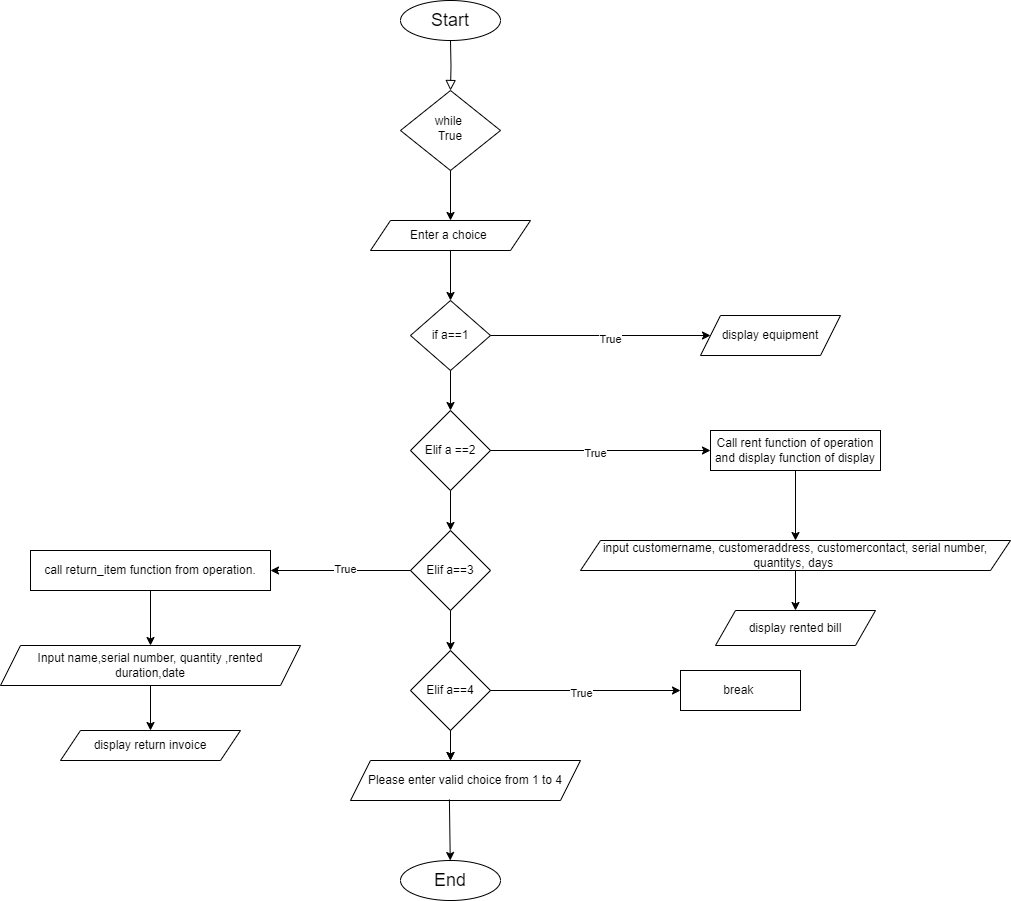
Step 6: If choose 4 end program

Step 7: Else display validation message

Step 8: Exit

### 2.2. Flowchart

Flowchart is a graphical representation of steps in a sequential order which is used to present the flow of process of program.



*Figure 1: Flowchart*

### 2.3. PseudoCode

**PseudoCode of Display.py**

CREATE function display\_table and pass item\_list in parameter

print (+----+------------------------+-----------+-------------------+------------+) print (|SN | Equipment Name | Brand │Price (5 days) │Quantity │)

print (+----+------------------------+-----------+-------------------+------------+) FOR each item in item\_list:

Print( serial number left-aligned within 4 spaces, name (up to 35 characters) left-aligned within 35 spaces, brand (up to 20 characters) leftaligned within 20 spaces, price right-aligned within 12 spaces, quantity rightaligned within 8 spaces) End For loop

print (:-----+-------------------------+----------+--------------------+-------------+)

End For loop

End Function

**PseudoCode of Write.py**

CREATE function write\_bill and pass filename, item\_list in parameter

OPEN a file name file\_name in write mode

START FOR LOOP

FOR each item in item\_list:

CREATE variable line = f"{item['serial\_number']},

{item['name']},{item['brand']},${item['price']}, {item ['quantity']}\n"

WRITE line in a file

END FOR LOOP

**PseudoCode of Main.py**

IMPORT display

IMPORT operation

IMPORT read

DECLEAR a function main()

CREATE item\_list which read data from items.txt using read\_item function

CREATE banner variable that contain banner

PRINT banner

While True:

PRINT enter 1 for display items

PRINT enter 2 for rent items

PRINT enter 3 for return items PRINT enter 4 for quit

TRY

CREATE variable a for user input

IF a == 1:

Call display\_table function with item\_list as argument

ELIF a == 2:

Call display\_table function with item\_list as argument Call rent function from operation and pass parameter

item\_list

ELIF a == 3:

Call return\_item function from operation and pass

Parameter item\_list

ELIF a == 4:

Print message "Thank you for trusting bot rental

service" BREAK

ELSE:

Print message "Please enter valid choose from 1 to 4"

EXCEPT Valueerror:

Print message"Please input as suggested"

Main()

**PseudoCode of Read.py**

CREATE function read\_item and pass filename in parameter

CREATE a item\_list as a list

OPEN a file named file\_name in read mode FOR each line in file:

CREATE a data variable that Split the line by comma and space to get pieces of data

CREATE a dictionary item that

SET'serial\_number' key to the integer value of data[0]

SET 'name' key to data[1]

SET 'brand' key to data[2]

Replace '$' from data[3] and convert it to a floatingpoint value, set as 'price' key

SET 'quantity' key to the integer value of data[4] ADD the item dictionary to the end of items\_ list

End FOR

RETURN the item\_ list

**PseudoCode of Operation.py**

IMPORT datetime IMPORT write

CREATE function rent and pass item\_list in parameter

CREATE customer name variable which ask user for name

CREATE customer address variable which ask user for address

CREATE customer contact with integer data type variable which ask user for number

CREATE serial\_number variable which Ask the user for the serial number of item

CREATE quantity variable with integer data type which ask user for quantity

CREATE duration variable with integer data type

CREATE item variable and call find\_item\_by\_serial\_number function

IF item is None then

Print serial number doesn't exists

RETURN

IF item quantity ==0 then

Print out of stock

RETURN

CREATE total\_amount variable that store item[price] \* quantity \* duration

CREATE vat variable that store round(.13 \* total\_amount)

CREATE grand\_total variable that store total\_amount + vat

CREATE rentdate variable that store variable

CREATE invoice variable that store bill details and create bill using f String

Decrease item quantity by use input quantity

Update the item\_list file using write\_bill function of write class

CREATE bill variable that open in write mode and generate rented bill

Print message"Rented successful"

Print invoice

Create function return\_equipment and pass filename, item\_list, in parameter

CREATE customer name variable which ask user for name

CREATE serial\_number variable which Ask the user for the serial number

of item to return

CREATE renturn\_date that stores date

CREATE item and call find\_item\_by\_serial\_number function

IF item is None then

Print messag " serial number doesn't"

Return

CREATE rental\_duartion variable with integer data type

CREATE fine variable and store 20

CREATE rental variable with integer data type

STORE the date in yyyy,mm,dd variable

CREATE a variable rentaldate that store date

CREATE a variable slow and store day

IF slow<= rental\_duration then

Day =0

ELSE

Day= slow-rental\_duration

CREATE a variable fine that store fine\_amt \* amt

CREATE a variable d that store rental\_duration/5 as a int If rental\_duartion /5 remainder =0 then

d=0+d

else

d=1+d

CREATE a variable total\_amount that calculate price of item \* rental\_duration+ rental duration

CREATE vat variable that store round(.13 \* total\_amount)

CREATE grand\_total variable that store total\_amount + vat + fine

CREATE invoice variable that store bill details and create bill using f String

Increase item quantity by use input quantity

Update the item\_list file using write\_bill function of write class

CREATE bill variable that open in write mode and generate rented bill

(replace spaces with underscores)

Print "Item return successfully !"

Print invoice

Create function find\_item\_by\_serial\_number and pass item\_list, serial\_number in parameter

FOR item in item\_List

IF serial\_number of item matches the given serial\_number then Return item

Return None

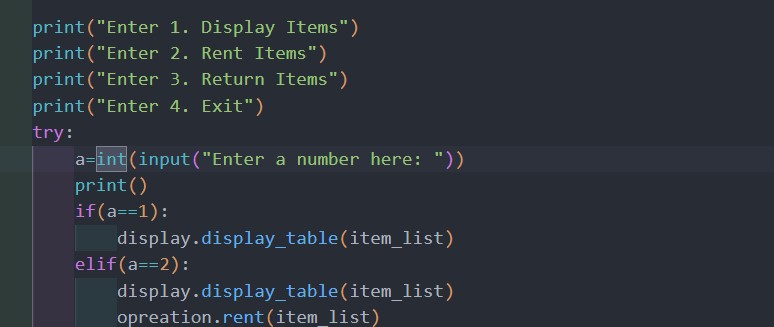
### 2.4. Data Structure

A data structure is an important tool for any coder. It is used to store data in memory in an organised manner while the programme is running. It explains the relationship between data and the logical operators that are applied to it. It is divided into two categories: primitive and non-primitive data types.

The data types which are used in this course are both primitive and non-primitive data types which are: -

1. Integer

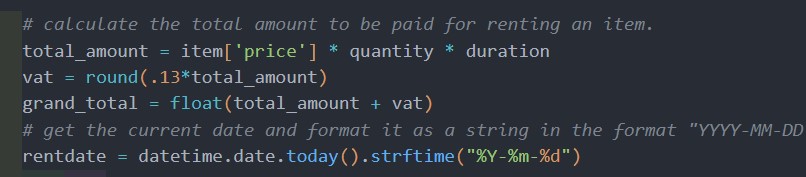
It is a numerical data types which holds the positive and negative numbers which does not hold any decimal points. Such as 1, -3, 100 etc. In this course work it is used to get user input .



*Figure 2: Integer Data Type*

1. Float

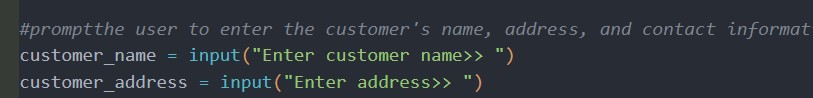
It is also a numerical data types but it holds the decimals numbers or rational numbers. In this course work it holds the value of grand total of rented item as shown in below



*Figure 3: Float Data Type*

1. String

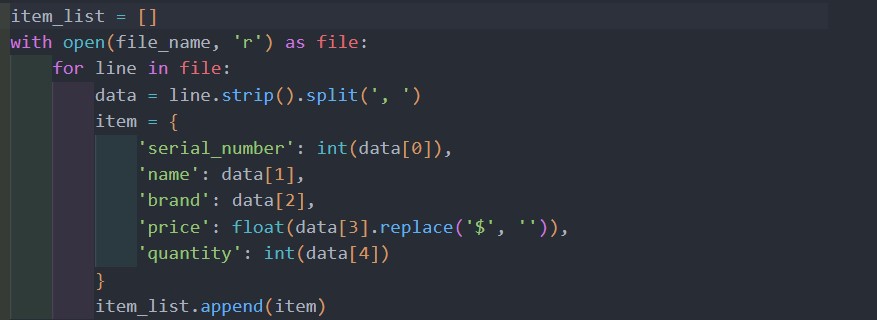
It is a collection if alphabetical word, which is used to create a single or multiple lines of world. In this course work it stores the decision of the user as shown in below



*Figure 4: String Data Type*

1. List

It is a non-primitive data type, it is versatile data structure which is written as a list of comma-separated elements enclosed with in square brackets. In this course work it is used to items.

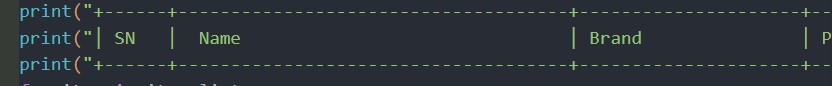


*Figure 5: List Data type*

## 3. Program

1. Display.py

In this module it show items in a table format Function use in display module are: print(): - this function is used to print the given output.

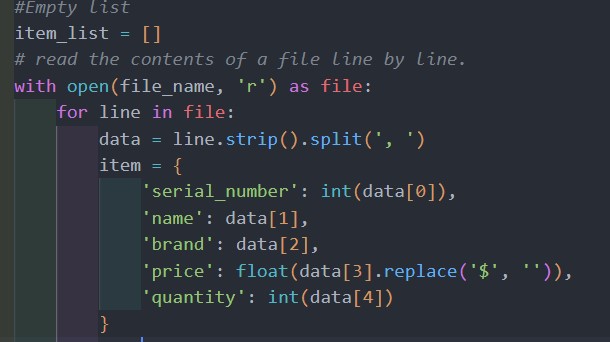


*Figure 6: Print Function*

1. Read.py

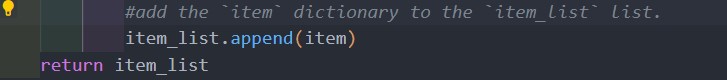
In this module the function read\_item read the content of file and add item in item\_list dictionary

read(): - this function is used to read the file.



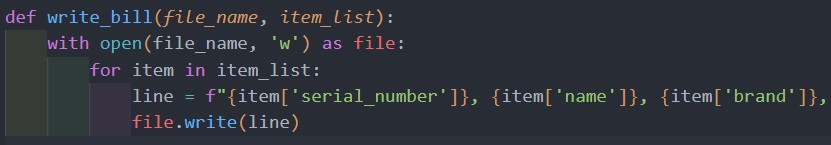
*Figure 7: Read Function*

append(): - this function is used to add elements to list.



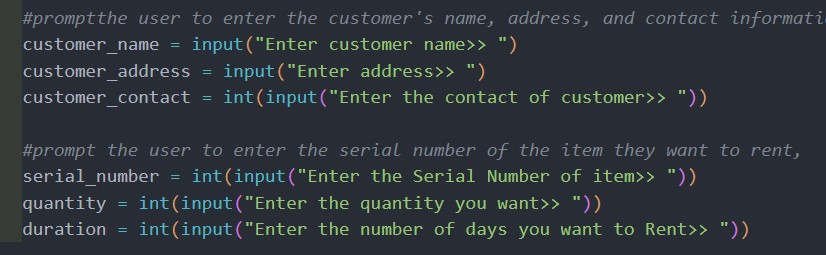
*Figure 8: Append Function*

1. Write.py write(): this function is used to write bill.



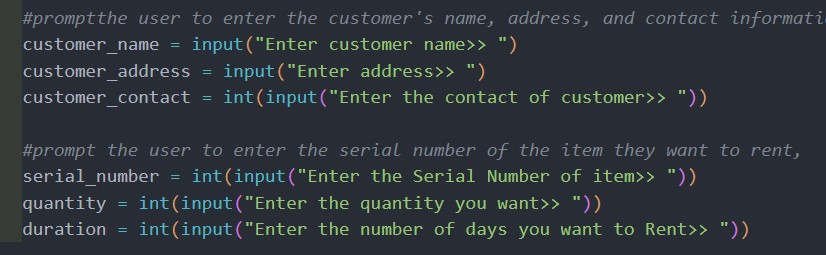
*Figure 9: Write function*

1. Operation.py input():- this function takes user input



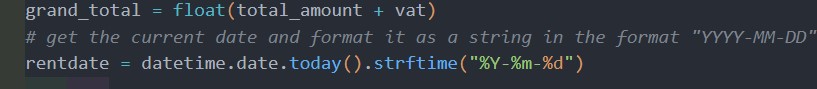
*Figure 10: Input Function*

int(): - this function is used to take integer number.



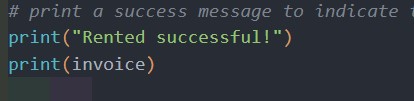
*Figure 11: Int Function*

float(): - this function is used to take float number.



*Figure 12: Float Function*

print() : - this function is used to print the standard output



*Figure 13: Print Function*

## 4. Testing

Testing is a crucial component of a program since it allows a programmer or developer to see if their program is working properly or not. If it isn't, it allows the developer to identify the problem and fix it.

**Test 1:**

*Table 1: Test 1 table*

|  |  |
| --- | --- |
| Objectives | To check whether it can handle exceptions or not |
| Action | Entering invalid input |
| Expected Output | It should be able to handle any sum of input of user |
| Actual Output | The program was able to handle even the wrong input of use |
| conclusion | The program runs in its flow until user closes it |



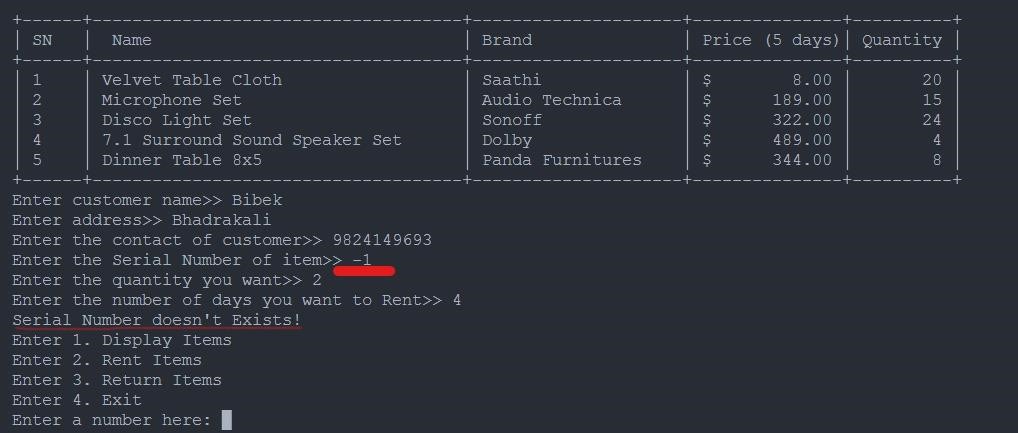
*Figure 14: Figure of Test 1:*

### Test 2

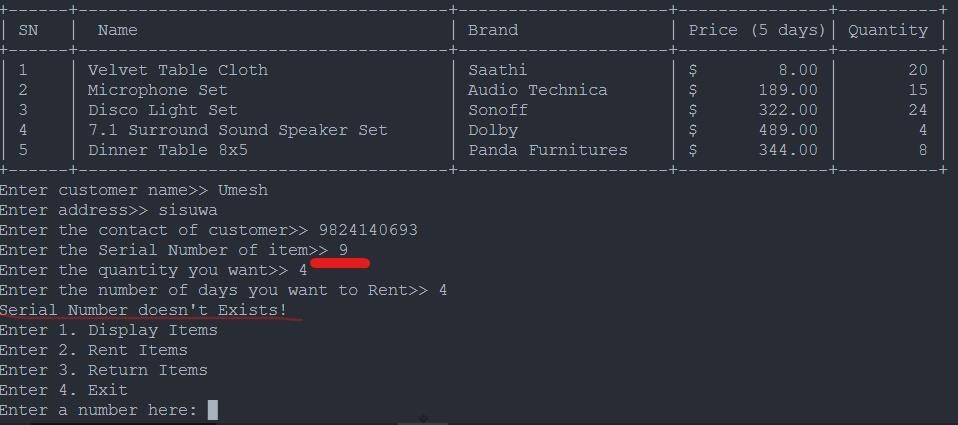
**Test of Rent:**

*Table 2:Test table of rent item*

|  |  |
| --- | --- |
| Objectives | To check whether it can accept negative value or non existing value |
| Action | Providing negative and non exsting value on rent |
| Expected Output | Program should display error message |
| Actual Output | Display error message |
| conclusion | Test successful |



*Figure 15: Enter negative value*

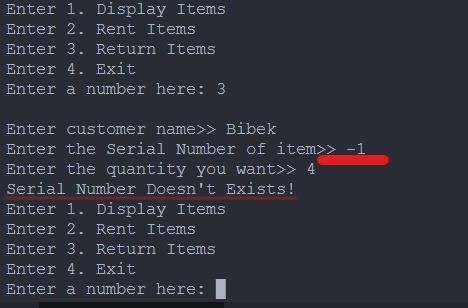


*Figure 16: Entering non existing value*

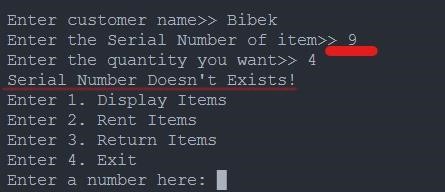
**Test of Return**

*Table 3: Test table of return item*

|  |  |
| --- | --- |
| Objectives | To check whether it can accept negative value or non existing value |
| Action | Providing negative and non exsting value on return |
| Expected Output | Program should display error message |
| Actual Output | Display error message |
| conclusion | Test successful |



*Figure 17: Entering negative value in return items*

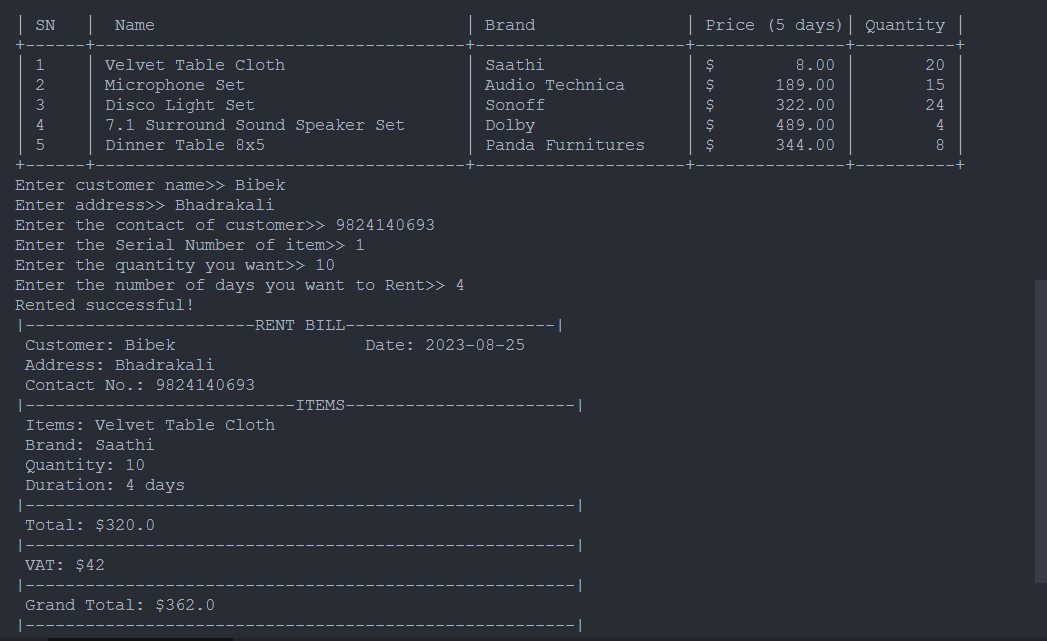


*Figure 18: Entering non existing value on return item*

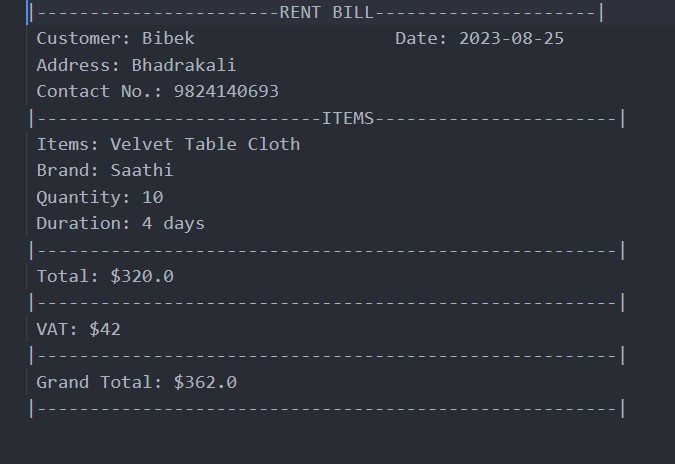
### Test 3

*Table 4: Test 3 table of rent bill*

|  |  |
| --- | --- |
| Objectives | To rent the available item and show the bill generated in text file and shell. |
| Action | Run the renting process and enter all the entered values |
| Expected Output | The program should run smoothly and generate bill in text file and display in shell |
| Actual Output | The program run smoothly and generate bill in text file and display in shell |
| conclusion | Test successful |



*Figure 19: Rent bill shown on shell*

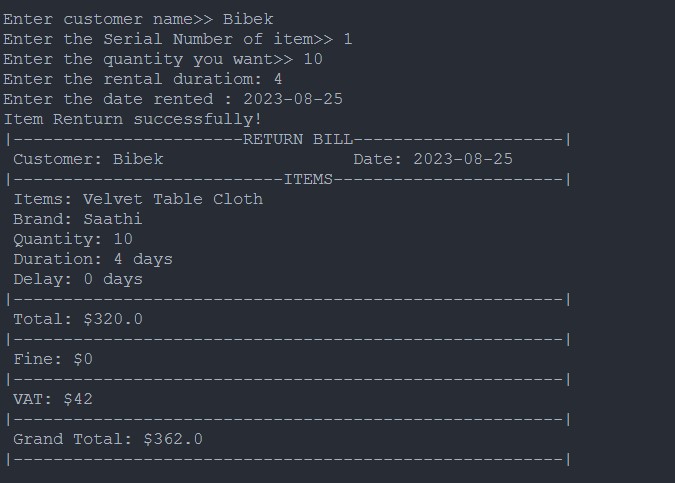


*Figure 20: Rented Bill*

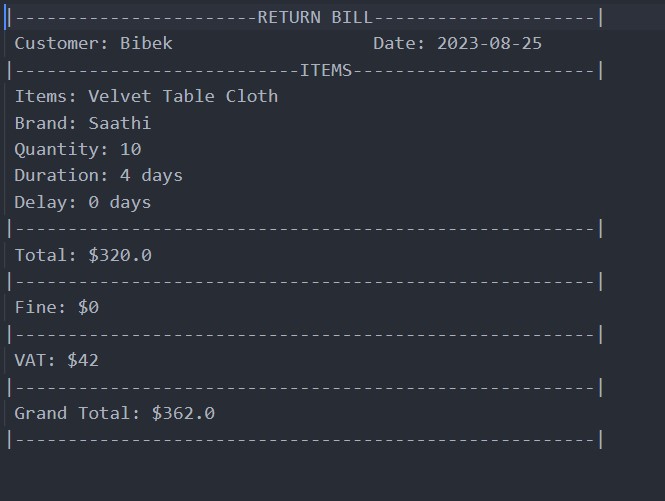
### Test 4

*Table 5: Test 4 table*

|  |  |
| --- | --- |
| Objectives | To rent the a rented item and show the return bill generated in text file and shell. |
| Action | Run the returning process and enter all the entered values |
| Expected Output | The program should run smoothly and generate return bill bill in text file and display in shell |
| Actual Output | The program run smoothly and generate return bill in text file and display in shell |
| conclusion | Test successful |



*Figure 21: Return bill shown on shell*

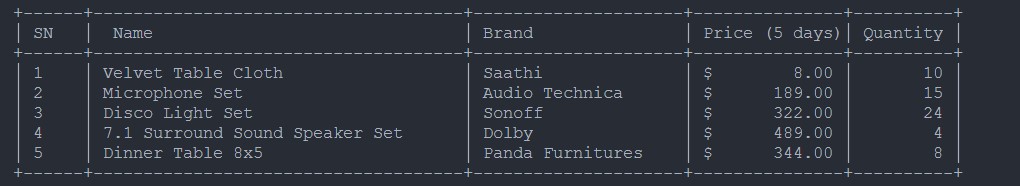


*Figure 22: Returned Bill*

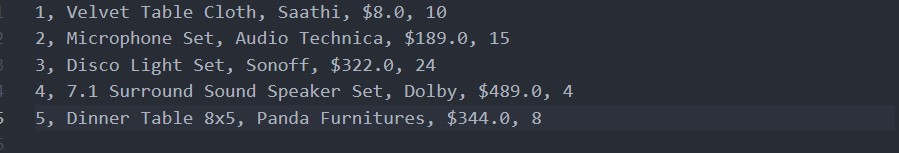
### Test 5

*Table 6: Table of test 5*

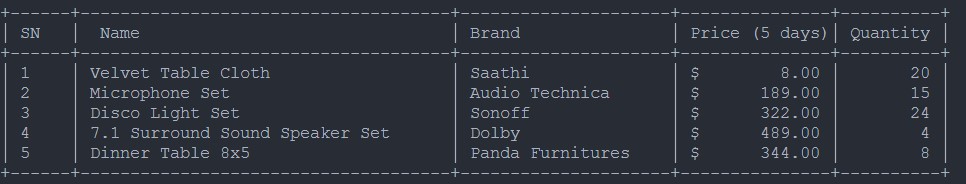
|  |  |
| --- | --- |
| Objectives | Updated quantity according to return and rent |
| Action | Run the renting and returning process successfully |
| Expected Output | The quantity should be updated accordingly to quantity rented and returned |
| Actual Output | The quantity should be updated accordingly to quantity rented and returned |
| Conclusion | Test successful |



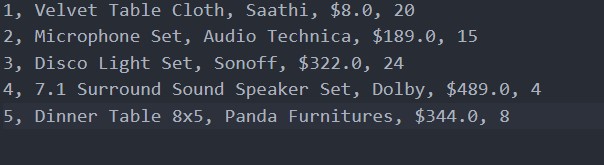
*Figure 23: After renting equipment*



*Figure 24: Updated txt after renting item*



*Figure 25: After returning items*



*Figure 26: After returning item txt updated file*

## 5. Conclusion

This course work is accomplished with numerous errors and corrections, as well as assistance and guidance.The module leader's assistance. There were several parts of this course that confused me, therefore the module leader provided pointers and support to help me understand to complete this course works. I also conducted extensive study to accomplish this course task on the subject of algorithms, flowcharts, pseudocode, and so on.

This course work helped me comprehend the program lot better while also learning more about Python, its built-in functions, comments, while and for loops, if/else conditions, and many other things. This course work was finished with the help and support of the module leader and hard work; during the course work completion, there was enthusiasm and thrill to finish the course work on time and to do better.

The coursework's major goal and objective is to create a system that will assist with create a billing system that is automated. We can print it on paper and give it to the appropriate person. Python is being used by customers. It is also intended to assess the student's general understanding of the module.It was a test to see if we could use suitable datatypes and functions to make the system work function properly. We can create any system like this if we have the information. The key goal was to understand how a system operates as well as its overall use and functionality of python is a language.

## 6. Bibliography

UpGrad, 2020. *upGrad.* [Online]

Available at: https://www.upgrad.com/blog/data-structures-algorithm-in-python/ [Accessed 21 Aug 2023].

Visual Studio Code , 2023. *Visual Studio Code.* [Online]

Available at: https://code.visualstudio.com/docs/editor/whyvscode

[Accessed 21 Aug 2023].

## 7. Appendix

### Display.py

def display\_table(item\_list): print("+------+-------------------------------------+---------------------+---------------+----------+")

print("│ SN │ Name │ Brand │ Price (5 days)│ Quantity

│") print("+------+-------------------------------------+---------------------+---------------+----------+") for item in item\_list:

print("│ {:<4} │ {:<35} │ {:<20}│ ${:>12.2f} │ {:>8} │".format( item['serial\_number'], item['name'][:35], item['brand'][:20], item['price'], item['quantity'])) print("+------+-------------------------------------+---------------------+---------------+----------+")

### Read.py

#The function reads the contents of a file. def read\_item(file\_name):

#Empty list item\_list = []

# read the contents of a file line by line. with open(file\_name, 'r') as file: for line in file:

data = line.strip().split(', ') item = {

'serial\_number': int(data[0]),

'name': data[1],

'brand': data[2],

'price': float(data[3].replace('$', '')),

'quantity': int(data[4])

}

#add the `item` dictionary to the `item\_list` list. item\_list.append(item) return item\_list

### Write.py

#Creatw write bill function def write\_bill(file\_name, item\_list):

with open(file\_name, 'w') as file: for item in item\_list:

line = f"{item['serial\_number']}, {item['name']}, {item['brand']}, ${item['price']},

{item['quantity']}\n" file.write(line)

### Operation.py

import datetime from datetime import date import write

def rent(item\_list):

#promptthe user to enter the customer's name, address, and contact information. customer\_name = input("Enter customer name>> ") customer\_address = input("Enter address>> ")

customer\_contact = int(input("Enter the contact of customer>> "))

#prompt the user to enter the serial number of the item they want to rent, serial\_number = int(input("Enter the Serial Number of item>> ")) quantity = int(input("Enter the quantity you want>> ")) duration = int(input("Enter the number of days you want to Rent>> "))

# call the `find\_item\_by\_serial\_number` function and passing the `item\_list` and `serial\_number` as

# arguments and find the item in the `item\_list` that matches the given `serial\_number` and assign it to the variable `item`.

item = find\_item\_by\_serial\_number(item\_list, serial\_number)

#check if the variable `item` is `None` then the item with the given serial number was not found in the `item\_list` and it prints "Items not found!" and returns from the function.

# handle the case when the user enters an invalid serial number for an item. if item is None:

print("Serial Number doesn't Exists!") return

# check if the quantity of the item in stock is equal to the quantity requested by the user

# If they are equal if item['quantity'] == 0: print("Out of stock!") return

# calculate the total amount to be paid for renting an item.

total\_amount = item['price'] \* quantity \* duration vat = round(.13\*total\_amount)

grand\_total = float(total\_amount + vat)

# get the current date and format it as a string in the format "YYYY-MM-DD" rentdate = datetime.date.today().strftime("%Y-%m-%d")

#Bill of rented Items invoice = f"|-----------------------RENT BILL---------------------|\n" \ f" Customer: {customer\_name} Date: {rentdate}\n" \ f" Address: {customer\_address}\n" \ f" Contact No.: {customer\_contact}\n" \ f"|---------------------------ITEMS-----------------------|\n" \ f" Items: {item['name']}\n" \ f" Brand: {item['brand']}\n" \ f" Quantity: {quantity}\n" \ f" Duration: {duration} days\n" \ f"|-------------------------------------------------------|\n" \ f" Total: ${total\_amount}\n" \ f"|-------------------------------------------------------|\n" \ f" VAT: ${vat}\n"\ f"|-------------------------------------------------------|\n" \ f" Grand Total: ${grand\_total}\n" \

f"|-------------------------------------------------------|\n" \

#Decrease the quantity of the rented item from the total quantity available in the item list.

item['quantity'] -= quantity write.write\_bill("items.txt", item\_list)

#creating a file name for the rental bill based on the customer's name bill = f"{customer\_name.replace(' ', '\_')}\_rental\_bill.txt"

with open(bill, 'w') as file:

file.write(invoice)

# print a success message to indicate that the rental process was completed successfully. print("Rented successful!") print(invoice)

#Create function rent\_item def return\_item(item\_list):

#prompt the user to enter the customer's name, the serial number of the item they # want to rent, and the quantity of the item they want to rent. customer\_name = input("Enter customer name>> ") serial\_number = int(input("Enter the Serial Number of item>> ")) quantity = int(input("Enter the quantity you want>> ")) return\_date = datetime.date.today().strftime("%Y-%m-%d")

item = find\_item\_by\_serial\_number(item\_list, serial\_number)

#check if the variable `item` is `None` then the item with the given serial number was not found in the `item\_list` and it prints "Items not found!" and returns from the function.

# handle the case when the user enters an invalid serial number for an item. if item is None:

print("Serial Number Doesn't Exists!") return

rental\_duration = int(input("Enter the rental duratiom: ")) fine\_amt = 20 rental= input("Enter the date rented : ").split('-')

yyyy,mm,dd=[int(item) for item in rental] rentalDate=date(yyyy,mm,dd) slow = (datetime.date.today() - rentalDate).days

# check if the number of days the item was rented is less than or equal to therental duration if slow<=rental\_duration:

day=0 else:

day=slow-rental\_duration

# calculating the fine amount for returning the rented item late. fine = fine\_amt \* day\*quantity

# calculating the number of weeks (d) that the item was rented for.

d=int(rental\_duration/5)

if rental\_duration %5 ==0:

d +=0 else: d +=1

# calculate the total amount to be paid for renting an item. total\_amount = item['price'] \* quantity \* rental\_duration

vat = round(.13\*total\_amount) grand\_total = float(total\_amount + vat + fine)

#Bill of rented Items invoice = f"|-----------------------RETURN BILL---------------------|\n" \ f" Customer: {customer\_name} Date: {return\_date}\n" \ f"|---------------------------ITEMS-----------------------|\n" \ f" Items: {item['name']}\n" \ f" Brand: {item['brand']}\n" \ f" Quantity: {quantity}\n" \ f" Duration: {rental\_duration} days\n" \ f" Delay: {day} days\n" \ f"|-------------------------------------------------------|\n" \ f" Total: ${total\_amount}\n" \ f"|-------------------------------------------------------|\n" \ f" Fine: ${fine}\n" \ f"|-------------------------------------------------------|\n" \ f" VAT: ${vat}\n"\ f"|-------------------------------------------------------|\n" \ f" Grand Total: ${grand\_total}\n" \ f"|-------------------------------------------------------|\n" \

#Increase the quantity of the rented item from the total quantity available in the item list.

item['quantity'] += quantity write.write\_bill("items.txt", item\_list) #creating a file name for the return bill based on the customer's name bill = f"{customer\_name.replace(' ', '\_')}\_return\_bill.txt" with open(bill, 'w') as file:

file.write(invoice)

# print a success message to indicate that the rental process was completed successfully. print("Item Renturn successfully!") print(invoice)

#The function finds an item in a list based on its serial number. def find\_item\_by\_serial\_number(item\_list, serial\_number):

# iterated over each item in the `item\_list` and checking if the `serial\_number` of the item matches the given `serial\_number`. for item in item\_list: if item['serial\_number'] == serial\_number:

return item return None