



 slington 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:

Group:

London Met ID:

College ID:

Assignment Due Date:

Assignment Submission Date:

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

Table of Contents

1) INTRODUCTION	1
2) Discussion and Analysis	2
2.1)Algorithm.	2
2.2) Pseudocode.....	3
3) Data structure	21
4) Testing.....	22
5) CONCLUSION.....	38
6) APPENDIX	39
7) REFERENCES.....	47

List of Figures

Figure 1: Test 1 figure	23
Figure 2 Test 2 fig a	25
Figure 3 Test 2 fig b	26
Figure 4 Test 3a	28
Figure 5 Test 3b	29
Figure 6 Test 3c	30
Figure 7 Test 4 fig a	31
Figure 8 Test 4 fig b	32
Figure 9 Test 5.1 fig a	33
Figure 10 Test 5.1 fig b	34
Figure 11 Test 5.1 fig c.....	34
Figure 12 Test 5.2 fig a	35
Figure 13 Test 5.2 fig b	36
Figure 14 Test 5.2 fig c.....	37

List of Tables

Table 1 Test 1	22
Table 2 Test 2	24
Table 3 Test 3	27
Table 4 Test 4	30
Table 5 Test 5.1	33
Table 6 Test 5.2	35

1) INTRODUCTION

The program is written in python language. Python is a simple and easy to use and understanding language which feels like reading simple english. This pseudo code nature of python makes it easy to learn and understandable by beginners. Python is a general purpose language meaning it can be used to create a variety of different programs. It has simple syntax same to english language It has syntax that helps developers to write programs in a shortest lines compare to other programming language. It is also known for easiest programming language for beginners to learn.

It is used for developing websites and software, data analysis and task automation. It takes less development time and open source It is high level language portable works on linux, windows and mac fun to work with. Python can be easily installed from python.org when click on the download button python can be installed might after you complete the setup by executing the file for your platform.

Objective and Goals of Project

This coursework concept can also be found in real-life stock market. In a proper or record list. The record list keeps track of the costumes that are rented and returned. Different costumes are used to record different transactions, and it takes time to retrieve the exact record of when a costumes was rented when it is returned. As a result, this program can be extremely beneficial to the market. People should adapt to their surroundings. Because new technologies are being launched every day. The old methods are gradually being replaced by modern methods. This tool will assist market in keeping accurate records of costumes in their collections, as well as costumes rented and returned by customers. The application is simple to use and it saves time.

2) Discussion and Analysis

2.1)Algorithm.

- Step 1: Start
- Step 2: Display if the user want to rent or return the costume or exit the program
- Step 3: Take input from user if to rent "1" or for return "2" and for exit "3"
- Step 4: Enter the costume id which want to rent
- Step 5: It checks for the costume is available or not.
- Step 5: Enter the name of the customer who want to rent the costume
- Step 6: Enter if user want to rent more costume
- Step 7: The costume was successfully rented
- Step 8: Enter the name of the customer who rented the costume
- Step 9: Enter the ID of the costume wanted to return
- Step 10: The costume was successfully returned
- Step 11: Enter the billing process
- Step 12: Enter how many days it has rented
- Step 13: Enter x for more costumes to return. Enter n if it is not
- Step 14: Stop

2.2) Pseudocode.

Pseudocode for Cli.py

FROM Rental IMPORT rent_item

FROM Rental IMPORT RETURNItems

FROM Rental IMPORT parse_invoice

DEFINE FUNCTION RETURN_item():

 filename=INPUT("Enter the invoice id: ").strip()

 fileItem=RETURNItems(filename)

 IF fileItem==None:

 RETURN

 IF fileItem=="not paid":

 OUTPUT("Invoice not paid")

 IF type(fileItem)==str:

 OUTPUT(""*10+"Error"+""*10)

 OUTPUT(fileItem)

 RETURN

 parse_invoice(fileItem)

```
DEFINE FUNCTION interact():
```

```
    WHILE True:
```

```
        OUTPUT("\n")
```

```
        OUTPUT("-"*25+"Welcome to Store"+"-"*25)
```

```
        OUTPUT("Press 1 to rent item")
```

```
        OUTPUT("Press 2 to RETURN item")
```

```
        OUTPUT("Press 3 to exit store")
```

```
        userInput=0
```

```
        WHILE True:
```

```
            TRY:
```

```
                userInput=int(INPUT("\nEnter your choice:"))
```

```
                break
```

```
            EXCEPT:
```

```
                OUTPUT("Choose from 1-3")
```

```
                continue
```

```
        IF userInput==1:
```

```
            rent_item()
```

```
        IF userInput==2:
```

```
    RETURN_item()
```

```
    IF userInput==3:
```

```
        break
```

```
    OUTPUT("Thanks come again!")
```

```
DEFINE FUNCTION main():
```

```
    interact()
```

Pseudocode for Datetime.py.

FROM datetime IMPORT datetime

IMPORT sys

FROM Rental.rentConfig IMPORT STORE_STOCK

FROM Rental.RETURNItem IMPORT parse_invoice, readInvoice

DEFINE CLASS OutOfStockException(Exception):

...

DEFINE FUNCTION read_stocks()->dict:

stock={}

TRY:

SET file TO open(STORE_STOCK, "r")

except FileNotFoundError as e:

OUTPUT(e)

RETURN {}

FOR i IN file.read().strip().split("\n"):

```
SET stocks TO i.strip().split(",")

SET stock[stocks[0].strip()] TO [x.strip() FOR x IN stocks[1:]]

file.close()

RETURN stock
```

```
DEFINE FUNCTION rentItems(customerName,
userOrder,noOfDays,amountPaid=0):

    stock=read_stocks()

    invioceld=datetime.timestamp(datetime.now())

    IF stock EQUALS {}:

        OUTPUT("No Stock found")

        sys.exit()

    SET totalPrice TO 0

    [totalPrice:=totalPrice+int(stock[str(i[0])][1].split("$")[1])*i[1] FOR i IN
userOrder]

    WHILE True:

        TRY:

            amountPaid=int(INPUT("Total is %s enter the advance amount you
want to pay ! "%(totalPrice*noOfDays)))
```

IF amountPaid < 0 :

 OUTPUT("Advance can not be less than 0")

IF amountPaid > (totalPrice*noOfDays):

 OUTPUT("You cannot pay more than
%s"%(totalPrice*noOfDays))

 continue

break

EXCEPT:

 OUTPUT("Invalid price please try again")

 continue

SET fileName TO "./invoice/"+str(inviocId).split(".")[0]+".txt"

SET file TO open(fileName, "w")

file.write("Customer Name: %s \n" % (customerName))

file.write("Invoice Id: %s \n" % (int(inviocId)))

file.write("Rented Date: %s \n" %

 (datetime.fromtimestamp(inviocId)).strftime("%Y/%m/%d %H-
%M-%S"))

file.write("Return Date: %s \n" %

```
(datetime.fromtimestamp(invioceld+noOfDays*24*60*60)).strftime("%Y/%m/%d %H-%M-%S"))
```

```
file.write("Total Amount: $%s \n" % (totalPrice*noOfDays))
```

```
file.write("Paid: %s \n"%amountPaid)
```

```
file.write("Returned: %s \n"%false)
```

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

```
FOR i IN userOrder:
```

```
    SET bookid TO str(i[0])
```

```
    file.write(bookid+", "+", ".join(stock[bookid][: -1]) + f", {i[1]} \n")
```

```
file.close()
```

```
RETURN str(invioceld).split(".")[0]
```

```
DEFINE FUNCTION updateStore(stock:dict):
```

```
    TRY:
```

```
        SET file TO open(STORE_STOCK, "w")
```

```
    except Exception as e:
```

```
        OUTPUT("No stock file found exiting the program")
```

```
        sys.exit()
```

```
    FOR i IN stock.keys():
```

```

file.write(

    ", ".join(str(x) FOR x IN [i, stock[i][0], stock[i][1], stock[i][2]])+"\n")

file.close()

```

DEFINE FUNCTION rent_item():

```

customerName=""

customerChoice=[]

stock=read_stocks()

WHILE True:

    OUTPUT("-"*60+"\n")

    OUTPUT("SN"+"    "*10+"Name"+"    "*25+"Price"+"    "*12+"Qunatity"+"
"*10)

    FOR key,value IN stock.items():

        OUTPUT(key+"    "*10+str(value[0])+"    "(30-
len(str(value[0]))) +str(value[1])+"    "(20-len(str(value[1]))) +str(value[2]))

    OUTPUT("-"*60+"\n")

    WHILE True:

        TRY:

```



```
id=int(INPUT("\nEnter your choice: "))

IF int(stock[str(id)][2])<1:

    OUTPUT("No stock available ")

    continue

WHILE True:

    quantity=int(INPUT("\nEnter the quantity: "))

    IF quantity < 1:

        OUTPUT("Quantity must be mre than 0")

    IF quantity > int(stock[str(id)][2]):

        OUTPUT("Product less IN stock compared to stock IN
store")

        continue

    break

stock[str(id)]

customerChoice.append([id,quantity])

stock[str(id)][2]=int(stock[str(id)][2])-quantity

break

except ValueError:

    OUTPUT("Invalid INPUT try again")
```

```
        continue

    except KeyError:

        OUTPUT("Item corresponding to user INPUT not found please
try again")

        continue

    IF not customerName:

        customerName=INPUT("\nEnter your Name: ")

        WHILE True:

            TRY:

                noOfDays=int(INPUT("\nEnter number of day you want to rent:
"))

                IF noOfDays >30:

                    OUTPUT("Can rent FOR 30 days only")

                    continue

                IF noOfDays < 1:

                    noOfDays=2

                    OUTPUT("Default is set to 2 day\n")

                    break

            EXCEPT:

                OUTPUT("Invalid INPUT")
```

continue

IF INPUT("Press y to rent another item again !").lower()=="y":
continue

break

invoiceId=rentItems(customerName,customerChoice,noOfDays)

updateStore(stock)

parse_invoice(readInvoice(invoiceId))

Psedocode for Return.py

IMPORT math

IMPORT sys

IMPORT Rental.rentItem as rent

IMPORT datetime

from Rental.rentConfig IMPORT INVOICE_PATH

DEFINE FUNCTION readInvoice(invoiceName):

'''

Read Invoice and stores it IN a dict

'''

SET invoiceDetail TO {}

SET invoiceDetail["items"] TO {}

TRY:

SET file TO open(INVOICE_PATH+invoiceName+".txt", "r")

EXCEPT:

RETURN {}

SET invoiceLists TO file.read().strip().split("\n\n\n\n\n\n")

```
SET invoiceHead TO invoiceLists[0]
```

```
SET invoiceBody TO invoiceLists[1]
```

```
file.close()
```

```
FOR i IN invoiceHead.strip().split("\n"):
```

```
    SET invoice TO i.strip().split(":")
```

```
    TRY:
```

```
        invoiceDetail[invoice[0].strip().replace(
```

```
            SET " ", "-").lower()] TO int(invoice[1].strip())
```

```
    EXCEPT:
```

```
        invoiceDetail[invoice[0].strip().replace(
```

```
            SET " ", "-").lower()] TO invoice[1].strip()
```

```
FOR i IN invoiceBody.strip().split("\n"):
```

```
    SET invoice TO i.strip().split(",")
```

```
        SET invoiceDetail["items"][invoice[0].strip()] TO [x.strip() FOR x IN  
invoice[1:]]
```

```
RETURN invoiceDetail
```

```
DEFINE FUNCTION parse_invoice(invoiceDetail):
```

```
    '''
```

```
        Parse the invoice dict IN console
```

```
    '''
```

```
    OUTPUT("-"*60+"\n")
```

```
    OUTPUT("-"*25+"Invoice FOR %s"%invoiceDetail["customer-name"]+"-
    "*25+"\n")
```

```
    FOR key,value IN invoiceDetail.items():
```

```
        IF type(value)==dict: continue
```

```
        OUTPUT(key.replace("-", " ").capitalize()+": "+str(value)+"\n")
```

```
    OUTPUT("SN"+" "*10+"Name"+" "*25+"Price"+" "*12+"Qunatity"+" "*10)
```

```
    FOR key,value IN invoiceDetail["items"].items():
```

```
        OUTPUT(key+" "*10+str(value[0])+" "*(30-
len(str(value[0])))+" "+str(value[1])+" "*(20-len(str(value[1])))+" "+str(value[2]))
```

```
    OUTPUT("-"*60+"\n")
```

```
DEFINE FUNCTION RETURNItems(filepath):
```

```
'''
```

```
    Checks the items, check delay and gives total price to the user
```

```
'''
```

```
SET stock TO rent.read_stocks()
```

```
SET invoiceDetail TO readInvoice(str(filepath))
```

```
TRY:
```

```
    IF (invoiceDetail["RETURNed"]!="false"):
```

```
        RETURN "Item already RETURNed"
```

```
except Exception as e:
```

```
    OUTPUT(e)
```

```
IF invoiceDetail EQUALS {}:
```

```
    OUTPUT("No invoice found of %s" % filepath)
```

```
    RETURN
```

```
    SET                                     RETURNDate                                     TO
    datetime.datetime.timestamp(datetime.datetime.strptime(
```

```
        invoiceDetail["RETURN-date"], "%Y/%m/%d %H-%M-%S"))
```

```
    SET                                     RETURNedDate                                     TO
    datetime.datetime.timestamp(datetime.datetime.now())
```

```
    SET          difference          TO          math.floor((RETURNedDate-
    RETURNDate)/(24*60*60))
```

```
SET fine TO 0
```

```
SET if difference <= 0: fine TO 0
```

```
SET else: fine TO difference*10
```

```
SET invoiceDetail["RETURNed-date"] TO  
datetime.datetime.now().strftime("%Y/%m/%d %H-%M-%S")
```

```
SET invoiceDetail["fine"] TO fine
```

```
SET amountToPay TO (
```

```
int(invoiceDetail["total-amount"].split("$")[1])-invoiceDetail["paid"])+fine
```

```
parse_invoice(invoiceDetail)
```

```
IF fine != 0:
```

```
    OUTPUT("There was %s day delay" % difference)
```

```
    OUTPUT("Fine of %s will be added to your pending amount" % fine)
```

```
OUTPUT("User needs to pay $%s" % amountToPay)
```

```
FOR i IN invoiceDetail["items"].keys():
```

```
    SET stock[i][2] TO int(stock[i][2])+int(invoiceDetail["items"][i][2])
```

```
IF (INPUT("Set invoice to paied ?").lower()=="y"):
```



```
rent.updateStore(stock)

updateInvoice(str(filepath), invoiceDetail)

RETURN invoiceDetail

RETURN "not paid"
```

DEFINE FUNCTION updateInvoice(filename, invoiceDetail: dict):

TRY:

```
SET file TO open(INVOICE_PATH+filename+".txt", "w")
```

EXCEPT:

```
OUTPUT("No stock file found exiting the program")
```

```
sys.exit()
```

```
SET invoiceDetail["paid"] TO int(
```

```
invoiceDetail["total-amount"].split("$")[1])+invoiceDetail["fine"]
```

```
invoiceDetail["RETURNed"]="true"
```

```
FOR key, value IN invoiceDetail.items():
```

```
IF type(value) EQUALS dict:
```

```
continue
```

```
file.write(key.replace("-", " ").capitalize()+" : "+str(value)+"\n")
```

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

```
FOR i IN invoiceDetail["items"].keys():
```

```
    file.write(
```

```
        ", ".join(str(x) FOR x IN [i, invoiceDetail["items"][i][0],  
invoiceDetail["items"][i][1], invoiceDetail["items"][i][2]])+"\n")
```

```
file.close()
```

```
parse_invoice(invoiceDetail)
```

3) Data structure

In Python, many data structures were used for input/output and data storage to complete a variety of tasks. Python offers a variety of data types, including integer, text, boolean, float, and more. Some data types and data structures were used when creating this software to store, manipulate, and carry out various actions on the data. This program uses a variety of different data kinds and structures, including integer, float, boolean, dictionary. These data types are objects in Python, thus they come with a ton of useful functions pre-built.

The different types of data which are used in this application are

- i) String
- ii) Integer
- iii) Boolean
- iv) Float
- v) Dictionary

4) Testing

Test no. 1	Discription
Objective	Testing implementation of try, except
Action	When we are asked to press an option and if we select an string value the program will say to put the numbers given in the option
Expected result	Program will ask us to choose from the given option saying Invalid Input
Actual result	Program asked us to choose from the given option saying Invalid Input
Conclusion	The test was successfull

Table 1 Test 1



```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\ACER\Desktop\pythonC (3)\pythonC\main.py =====

-----Welcome to Store-----
Press 1 to rent item
Press 2 to return item
Press 3 to exit store

Enter your choice:ab
Choose from 1-3

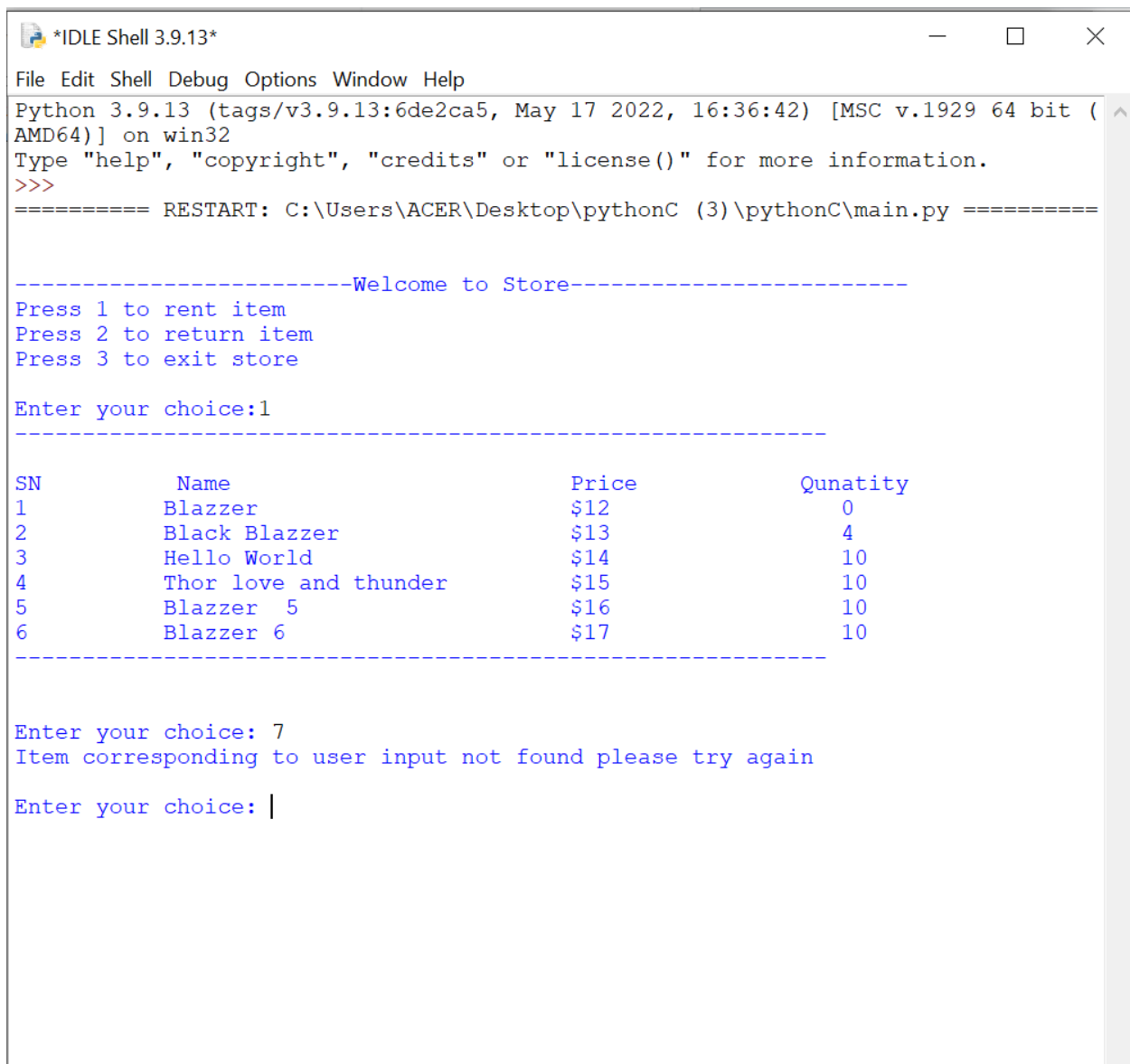
Enter your choice:|
```

Ln: 15 Col: 18

Figure 1: Test 1 figure

Test no. 2	Discription
Objective	Testing by providing negative value and non existed value
Action	When the program ask us to provide the no. of quantity of stock we want to rent or return And there if we put negative value or non existed value
Expected result	For negative value invalid input should appear and for non existed value we don't have that much quantity of stock should appear
Actual result	For negative value invalid input appeared and for non existed value we don't have that much quantity of stock appeared
Conclusion	The test was successfull

Table 2 Test 2



```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\ACER\Desktop\pythonC (3)\pythonC\main.py =====

-----Welcome to Store-----
Press 1 to rent item
Press 2 to return item
Press 3 to exit store

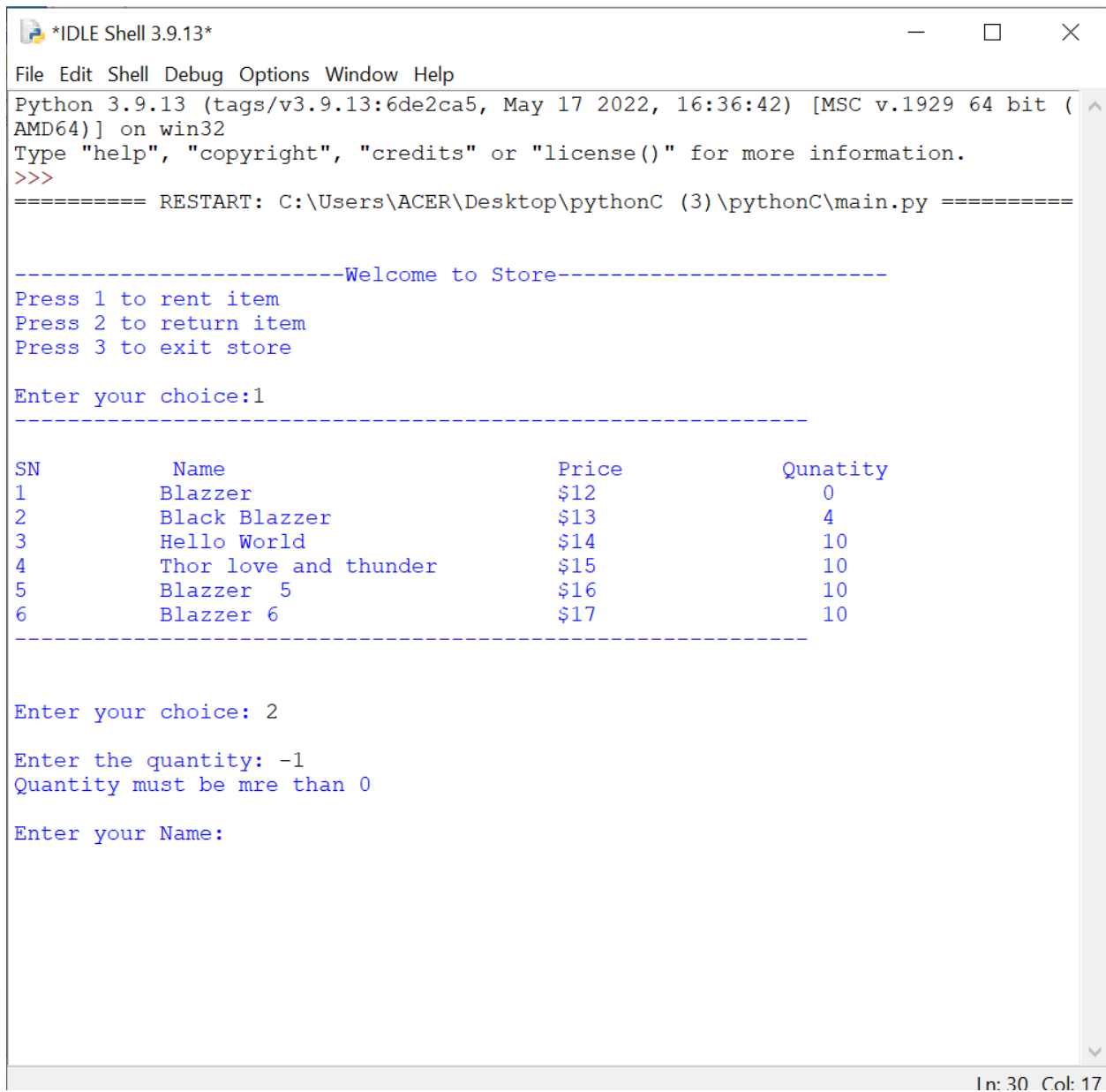
Enter your choice:1
-----

SN          Name          Price          Qunatity
1           Blazzer          $12            0
2           Black Blazzer    $13            4
3           Hello World      $14            10
4           Thor love and thunder $15            10
5           Blazzer 5        $16            10
6           Blazzer 6        $17            10
-----

Enter your choice: 7
Item corresponding to user input not found please try again

Enter your choice: |
```

Figure 2 Test 2 fig a



```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\ACER\Desktop\pythonC (3)\pythonC\main.py =====

-----Welcome to Store-----
Press 1 to rent item
Press 2 to return item
Press 3 to exit store

Enter your choice:1
-----

SN          Name          Price          Qunatity
1           Blazzer          $12             0
2         Black Blazzer          $13             4
3         Hello World          $14            10
4    Thor love and thunder          $15            10
5         Blazzer 5          $16            10
6         Blazzer 6          $17            10
-----

Enter your choice: 2

Enter the quantity: -1
Quantity must be mre than 0

Enter your Name:
```

In: 30 Col: 17

Figure 3 Test 2 fig b

Test 3

Test no. 3	Discription
Objective	Testing if the invoice txt file will get or not after renting
Action	After the completion of whole renting process which was done for multiple times Atlast a txt file of bill should be generated into the invoice folder
Expected result	A txt file of bill should be generated into the invoice folder
Actual result	A txt file of bill have been generated into the invoice folder
Conclusion	The test was successfull

Table 3 Test 3

```

*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help

-----Welcome to Store-----
Press 1 to rent item
Press 2 to return item
Press 3 to exit store

Enter your choice:1
-----

SN          Name          Price          Qunatity
1           Blazzer        $12            0
2           Black Blazzer  $13            4
3           Hello World    $14            0
4           Thor love and thunder $15            0
5           Blazzer 5       $16            10
6           Blazzer 6       $17            10
-----

Enter your choice: 5
Enter the quantity: 10
Enter your Name: Aayush Bam
Enter number of day you want to rent: 5
Press y to rent another item again !y
-----

SN          Name          Price          Qunatity
1           Blazzer        $12            0
2           Black Blazzer  $13            4
3           Hello World    $14            0
4           Thor love and thunder $15            0
5           Blazzer 5       $16            0
6           Blazzer 6       $17            10
-----

Enter your choice: 6

```

Figure 4 Test 3a

```
Enter your choice: 6
Enter the quantity: 10
Press y to rent another item again !n
Total is 1650 enter the advance amount you want to pay ! 1000
-----
-----Invoice for Aayush Bam-----
Customer name: Aayush Bam
Invoice id: 1661503723
Rented date: 2022/08/26 14-33-43
Return date: 2022/08/31 14-33-43
Total amount: $1650
Paid: 1000
Returned: false
SN      Name      Price      Qunatity
5       Blazzer 5    $16       10
6       Blazzer 6    $17       10
-----
-----Welcome to Store-----
Press 1 to rent item
Press 2 to return item
Press 3 to exit store
Enter your choice:|
```

Figure 5 Test 3b

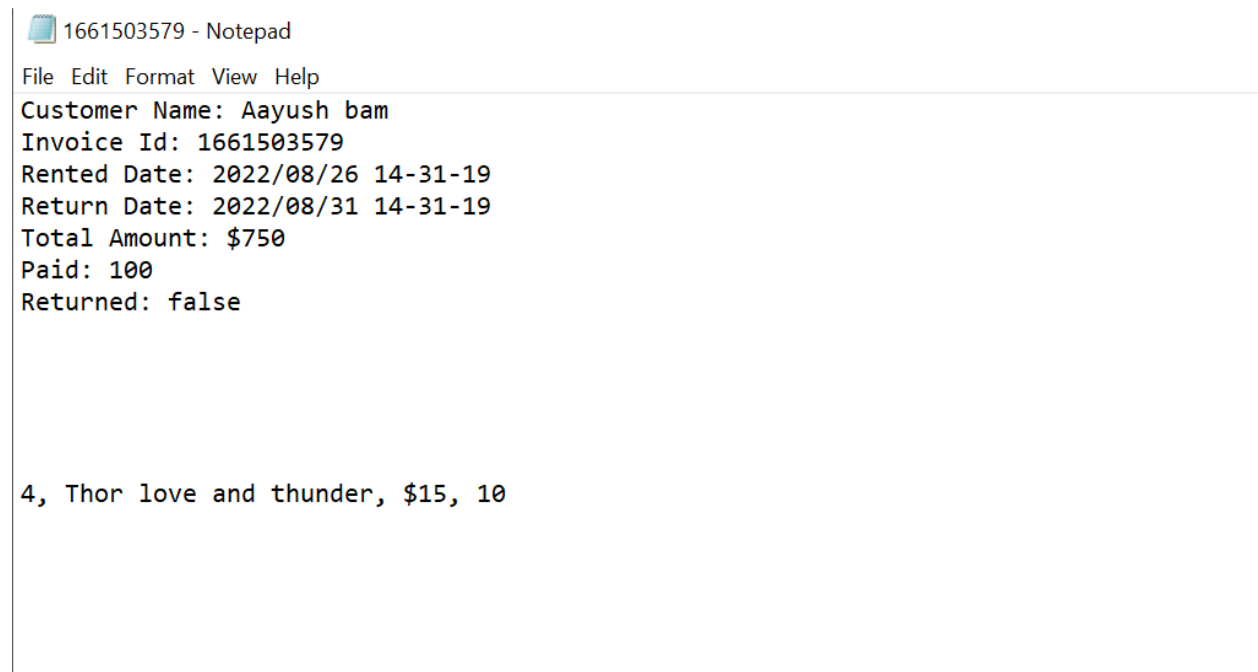


Figure 6 Test 3c

Test 4

Test no. 4	Description
Objective	Testing if the invoice txt file will get or not after returning
Action	After the completion of whole returning process which was done for multiple times At last a txt file of bill should be generated into the invoice folder
Expected result	A txt file of bill should be generated into the invoice folder after the completion of returning process
Actual result	A txt file of bill have been generated into the invoice folder after the completion of returning process
Conclusion	The test was successfull

Table 4 Test 4

```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
-----Welcome to Store-----
Press 1 to rent item
Press 2 to return item
Press 3 to exit store

Enter your choice:2
Enter the invoice id: 1661503723
-----

-----Invoice for Aayush Bam-----

Customer name: Aayush Bam
Invoice id: 1661503723
Rented date: 2022/08/26 14-33-43
Return date: 2022/08/31 14-33-43
Total amount: $1650
Paid: 1000
Returned: false
Returned date: 2022/08/26 14-35-09
Fine: 0

SN          Name          Price          Qunatity
5           Blazzer  5          $16            10
6           Blazzer  6          $17            10
-----

User needs to pay $650
```

Figure 7 Test 4 fig a

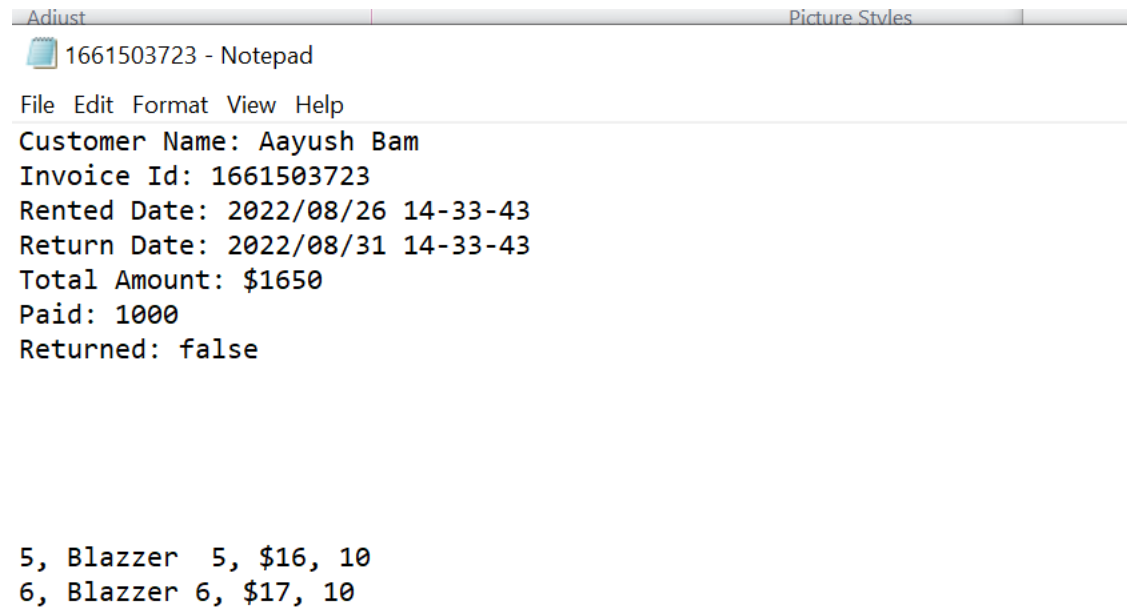
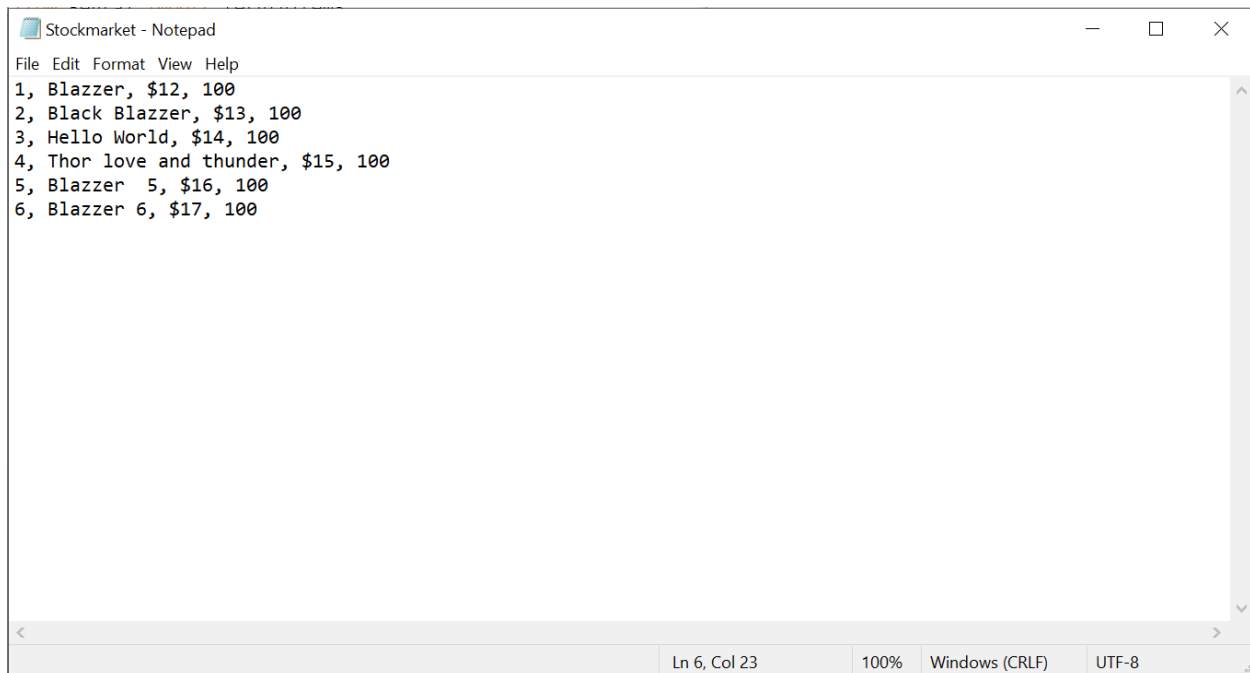


Figure 8 Test 4 fig b

Test 5.1

Test no. 5.1	Description
Objective	Testing if the stock file will get automatically updated after renting
Action	After the completion of whole renting process which was done by entering all the details after asked by the program Check if the file have been updated or not
Expected result	Stock should get automatically updated after whole renting process
Actual result	Stock got automatically updated after whole renting process
Conclusion	The test was successfull

Table 5 Test 5.1



```

Stockmarket - Notepad
File Edit Format View Help
1, Blazzer, $12, 100
2, Black Blazzer, $13, 100
3, Hello World, $14, 100
4, Thor love and thunder, $15, 100
5, Blazzer 5, $16, 100
6, Blazzer 6, $17, 100
Ln 6, Col 23 100% Windows (CRLF) UTF-8

```

Figure 9 Test 5.1 fig a

```

-----Welcome to Store-----
Press 1 to rent item
Press 2 to return item
Press 3 to exit store

Enter your choice:1
-----

SN      Name      Price      Qunatity
1      Blazzer      $12        100
2      Black Blazzer      $13        100
3      Hello World      $14        100
4      Thor love and thunder      $15        100
5      Blazzer 5      $16        100
6      Blazzer 6      $17        100
-----

Enter your choice: 1

Enter the quantity: 50

Enter your Name: Aayush

Enter number of day you want to rent: 5
Press y to rent another item again !no
Total is 3000 enter the advance amount you want to pay ! 2000
-----

-----Invoice for Aayush-----

Customer name: Aayush

Invoice id: 1661503952

Rented date: 2022/08/26 14-37-32

Return date: 2022/08/31 14-37-32

Total amount: $3000

Paid: 2000

```

Figure 10 Test 5.1 fig b

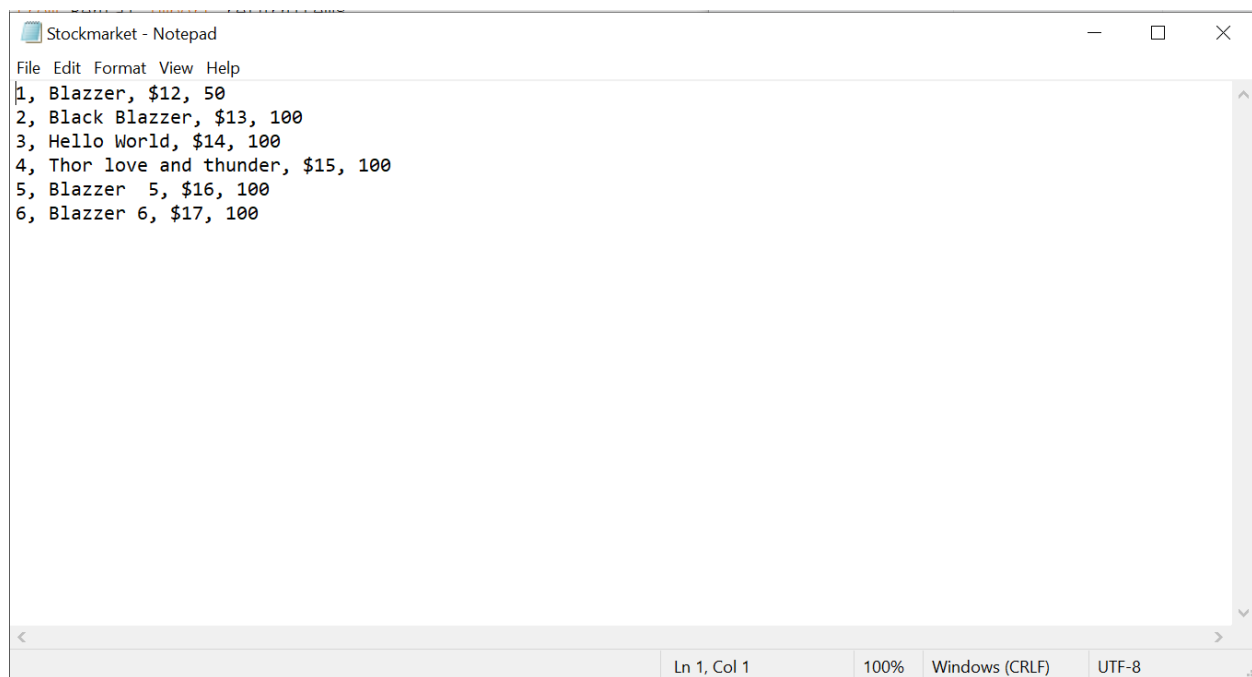
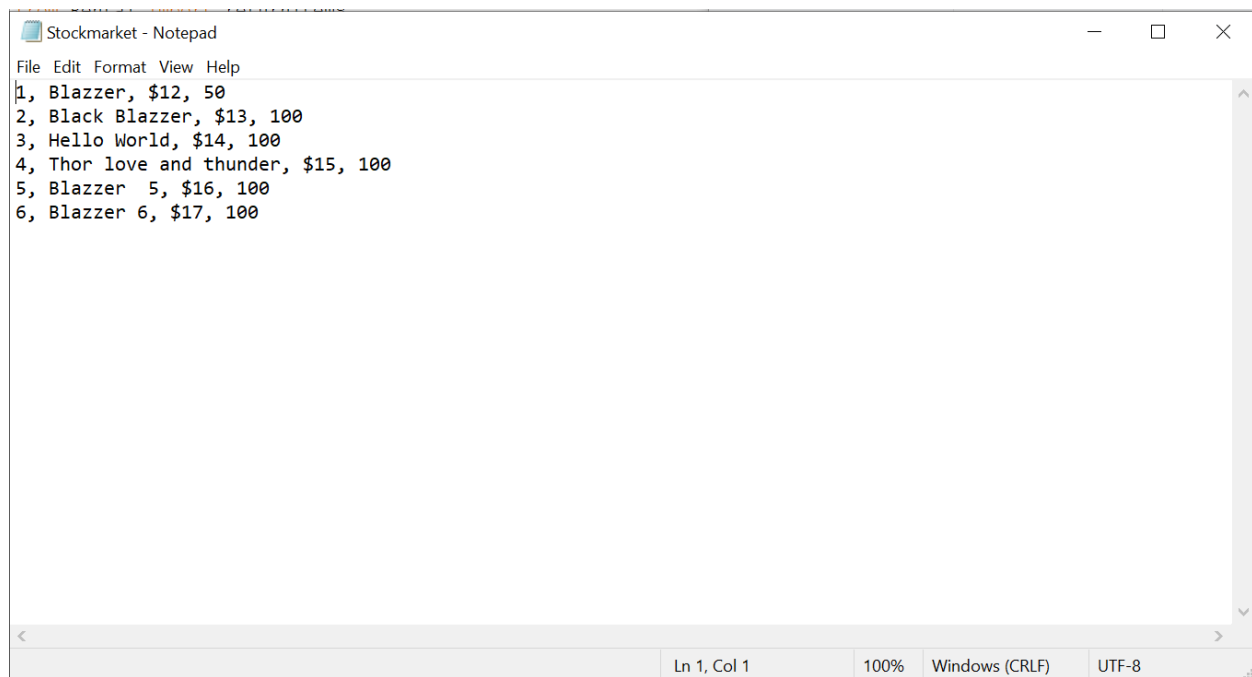


Figure 11 Test 5.1 fig c

Test no. 5.2	Description
Objective	Testing if the stock file will get automatically updated after returning
Action	After the completion of whole returning process which was done by entering all the details after asked by the program Check if the file have been updated or not
Expected result	Stock should get automatically updated after whole returning process
Actual result	Stock got automatically updated after whole returning process
Conclusion	The test was successfull

Table 6 Test 5.2



```

Stockmarket - Notepad
File Edit Format View Help
1, Blazzer, $12, 50
2, Black Blazzer, $13, 100
3, Hello World, $14, 100
4, Thor love and thunder, $15, 100
5, Blazzer 5, $16, 100
6, Blazzer 6, $17, 100
  
```

Figure 12 Test 5.2 fig a

```
-----Welcome to Store-----
Press 1 to rent item
Press 2 to return item
Press 3 to exit store

Enter your choice:2
Enter the invoice id: 1661503952
-----

-----Invoice for Aayush-----

Customer name: Aayush

Invoice id: 1661503952

Rented date: 2022/08/26 14-37-32

Return date: 2022/08/31 14-37-32

Total amount: $3000

Paid: 2000

Returned: false


Returned date: 2022/08/26 14-39-40

Fine: 0

SN          Name          Price          Qunatity
1           Blazzer          $12             50
-----

User needs to pay $1000
```

Figure 13 Test 5.2 fig b



```
1 main()
2
3 *Stockmarket - Notepad
4 File Edit Format View Help
5 1, Blazzer, $12, 100
6 2, Black Blazzer, $13, 100
7 3, Hello World, $14, 100
8 4, Thor love and thunder, $15, 100
9 5, Blazzer 5, $16, 100
10 6, Blazzer 6, $17, 100
```

The screenshot shows a Notepad window with the title bar '*Stockmarket - Notepad'. The menu bar includes File, Edit, Format, View, and Help. The text area contains a list of six items, each on a new line, starting with an ID number followed by a name, price, and quantity. The status bar at the bottom indicates 'Ln 6, Col 23', '100%', 'Windows (CRLF)', and 'UTF-8'.

Figure 14 Test 5.2 fig c

5) CONCLUSION

This coursework was incredibly helpful for learning, especially the practical application of file handling and data structure ideas. The tasks need a lot of vision and conception to be carried out effectively. The program helped students improve their knowledge of data structures and Python. There is no doubt that the data collected will be valuable in the future. In order to ensure that the software can be used in production or by a small group of people, this course also teaches students how to test it. I learned more about numerous Python-related topics that I previously knew about as well as some new ones through my research on various websites and articles.

The teachers taught us everything from how to type the codes to how to write the report. When we sent an email about the coursework, the professors/teachers replied quickly to the questions. I asked my friends about the issues I was having, and they helped me. I looked on the internet about items related to the coursework. At last, the coursework helps me in learning and putting me into practice, and helps me to improve my Python skills.

6) APPENDIX

6.1 Main module

```
from Rental import rent_item
from Rental import returnItems
from Rental import parse_invoice

def return_item():
    filename=input("Enter the invoice id: ").strip()
    fileItem=returnItems(filename)
    if fileItem==None:
        return
    if fileItem=="not paid":
        print("Invoice not paid")
    if type(fileItem)==str:
        print("***10+\"Error\"+\"**10)
        print(fileItem)
        return
    parse_invoice(fileItem)

def interact():
    while True:
        print("\n")
        print("-"*25+"Welcome to Store"+"-"*25)
        print("Press 1 to rent item")
```

```
print("Press 2 to return item")
print("Press 3 to exit store")
userInput=0
while True:
    try:
        userInput=int(input("\nEnter your choice:"))
        break
    except:
        print("Choose from 1-3")
        continue
if userInput==1:
    rent_item()
if userInput==2:
    return_item()
if userInput==3:
    break

print("Thanks come again!")

def main():
    interact()
```

6.2 Rent module

```
from datetime import datetime
```

```
import sys
```

```
from Rental.rentConfig import STORE_STOCK
```

```
from Rental.returnItem import parse_invoice, readInvoice
```

```
class OutOfStockException(Exception):
```

```
    ...
```

```
def read_stocks()->dict:
```

```
    stock={}
```

```
    try:
```

```
        file = open(STORE_STOCK, "r")
```

```
    except FileNotFoundError as e:
```

```
        print(e)
```

```
        return {}
```

```
    for i in file.read().strip().split("\n"):
```

```
        stocks = i.strip().split(",")
```

```
        stock[stocks[0].strip()] = [x.strip() for x in stocks[1:]]
```

```
    file.close()
```

```
    return stock
```

```
def rentItems(customerName, userOrder,noOfDays,amountPaid=0):  
    stock=read_stocks()  
    invioceld=datetime.timestamp(datetime.now())  
    if stock == {}:   
        print("No Stock found")  
        sys.exit()  
    totalPrice = 0  
    [totalPrice:=totalPrice+int(stock[str(i[0])][1].split("$")[1])*i[1] for i in userOrder]  
    while True:  
        try:  
            amountPaid=int(input("Total is %s enter the advance amount you want to pay !  
"%(totalPrice*noOfDays)))  
            if amountPaid <0 :  
                print("Advance can not be less than 0")  
            if amountPaid > (totalPrice*noOfDays):  
                print("You cannot pay more than %s"%(totalPrice*noOfDays))  
                continue  
            break  
        except:  
            print("Invalid price please try again")  
            continue  
    fileName = "./invoice/"+str(invioceld).split(".")[0]+".txt"  
    file = open(fileName, "w")  
    file.write("Customer Name: %s \n" % (customerName))
```



```

file.write("Invoice Id: %s \n" % (int(inviocId)))

file.write("Rented Date: %s \n" %

            (datetime.fromtimestamp(inviocId)).strftime("%Y/%m/%d %H-%M-%S"))

file.write("Return Date: %s \n" %

            (datetime.fromtimestamp(inviocId+noOfDays*24*60*60)).strftime("%Y/%m/%d %H-
%M-%S"))

file.write("Total Amount: $%s \n" % (totalPrice*noOfDays))

file.write("Paid: %s \n"%amountPaid)

file.write("Returned: %s \n"%false")

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

for i in userOrder:

    bookid = str(i[0])

    file.write(bookid+", "+", ".join(stock[bookid][:-1]) + f", {i[1]} \n")

file.close()

return str(inviocId).split(".")[0]

```

```

def updateStore(stock:dict):

    try:

        file = open(STORE_STOCK, "w")

    except Exception as e:

        print("No stock file found exiting the program")

        sys.exit()

    for i in stock.keys():

        file.write(

            ", ".join(str(x) for x in [i, stock[i][0], stock[i][1], stock[i][2]])+"\n")

```

```
file.close()
```

```
def rent_item():
    customerName=""
    customerChoice=[]
    stock=read_stocks()
    while True:
        print("-"*60+"\n")
        print("SN"+" "*10+"Name"+" "*25+"Price"+" "*12+"Qunatity"+" "*10)
        for key,value in stock.items():
            print(key+" "*10+str(value[0])+" "*(30-len(str(value[0])))
            +str(value[1])+" "*(20-len(str(value[1])))
            +str(value[2]))
        print("-"*60+"\n")
        while True:
            try:
                id=int(input("\nEnter your choice: "))
                if int(stock[str(id)][2])<1:
                    print("No stock available ")
                    continue
            while True:
                quantity=int(input("\nEnter the quantity: "))
                if quantity < 1:
                    print("Quantity must be mre than 0")
                if quantity > int(stock[str(id)][2]):
```

```
        print("Product less in stock compared to stock in store")
        continue
    break
    stock[str(id)]
    customerChoice.append([id,quantity])
    stock[str(id)][2]=int(stock[str(id)][2])-quantity
    break
except ValueError:
    print("Invalid input try again")
    continue
except KeyError:
    print("Item corresponding to user input not found please try again")
    continue
if not customerName:
    customerName=input("\nEnter your Name: ")
while True:
    try:
        noOfDays=int(input("\nEnter number of day you want to rent: "))
        if noOfDays >30:
            print("Can rent for 30 days only")
            continue
        if noOfDays < 1:
            noOfDays=2
            print("Default is set to 2 day\n")
        break
```

```
except:
    print("Invalid input")
    continue
if input("Press y to rent another item again !").lower()=="y": continue
break
invoiceId=rentItems(customerName,customerChoice,noOfDays)
updateStore(stock)
parse_invoice(readInvoice(invoiceId))
```

6.3 Return module

```

import math
import sys
import Rental.rentItem as rent
import datetime
from Rental.rentConfig import INVOICE_PATH

def readInvoice(invoiceName):
    """
    Read Invoice and stores it in a dict
    """
    invoiceDetail = {}
    invoiceDetail["items"] = {}
    try:
        file = open(INVOICE_PATH+invoiceName+".txt", "r")
    except:
        return {}
    invoiceLists = file.read().strip().split("\n\n\n\n\n")
    invoiceHead = invoiceLists[0]
    invoiceBody = invoiceLists[1]
    file.close()

    for i in invoiceHead.strip().split("\n"):
        invoice = i.strip().split(":")
        try:
            invoiceDetail[invoice[0].strip().replace(
                " ", "-").lower()] = int(invoice[1].strip())
        except:
            invoiceDetail[invoice[0].strip().replace(
                " ", "-").lower()] = invoice[1].strip()

    for i in invoiceBody.strip().split("\n"):
        invoice = i.strip().split(",")
        invoiceDetail["items"][invoice[0].strip()] = [x.strip() for x in invoice[1:]]
    return invoiceDetail

def parse_invoice(invoiceDetail):
    """
    Parse the invoice dict in console
    """
    print("-"*60+"\n")
    print("-"*25+"Invoice for %s"%invoiceDetail["customer-name"]+"-"+*25+"\n")

    for key,value in invoiceDetail.items():

```

```

        if type(value)==dict: continue
        print(key.replace("-", " ").capitalize()+": "+str(value)+"\n")
    print("SN"+" " *10+"Name"+" " *25+"Price"+" " *12+"Qunatity"+" " *10)
    for key,value in invoiceDetail["items"].items():
        print(key+" " *10+str(value[0])+" " *(30-len(str(value[0]))) +str(value[1])+" " *(20-
len(str(value[1]))) +str(value[2]))
    print("-"*60+"\n")

```

```

def returnItems(filepath):
    """
    Checks the items, check delay and gives total price to the user
    """
    stock = rent.read_stocks()
    invoiceDetail = readInvoice(str(filepath))
    try:
        if (invoiceDetail["returned"]!="false"):
            return "Item already returned"
    except Exception as e:
        print(e)
    if invoiceDetail == {}:
        print("No invoice found of %s" % filepath)
        return
    returnDate = datetime.datetime.timestamp(datetime.datetime.strptime(
        invoiceDetail["return-date"], "%Y/%m/%d %H-%M-%S"))
    returnedDate = datetime.datetime.timestamp(datetime.datetime.now())
    difference = math.floor((returnedDate-returnDate)/(24*60*60))
    fine = 0

    if difference <= 0: fine = 0
    else: fine = difference*10

    invoiceDetail["returned-date"] = datetime.datetime.now().strftime("%Y/%m/%d %H-
%M-%S")
    invoiceDetail["fine"] = fine
    amountToPay = (
        int(invoiceDetail["total-amount"].split("$")[1])-invoiceDetail["paid"])+fine
    parse_invoice(invoiceDetail)
    if fine != 0:
        print("There was %s day delay" % difference)
        print("Fine of %s will be added to your pending amount" % fine)
        print("User needs to pay $%s" % amountToPay)
    for i in invoiceDetail["items"].keys():
        stock[i][2] = int(stock[i][2])+int(invoiceDetail["items"][i][2])

    if (input("Set invoice to paied ?").lower()=="y"):

```

```
rent.updateStore(stock)
updateInvoice(str(filepath), invoiceDetail)
return invoiceDetail
return "not paid"
```

```
def updateInvoice(filename, invoiceDetail: dict):
    try:
        file = open(INVOICE_PATH+filename+".txt", "w")
    except:
        print("No stock file found exiting the program")
        sys.exit()
    invoiceDetail["paid"] = int(
        invoiceDetail["total-amount"].split("$")[1])+invoiceDetail["fine"]
    invoiceDetail["returned"]="true"
    for key, value in invoiceDetail.items():
        if type(value) == dict:
            continue
        file.write(key.replace("-", " ").capitalize()+": "+str(value)+"\n")

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

    for i in invoiceDetail["items"].keys():
        file.write(
            ", ".join(str(x) for x in [i, invoiceDetail["items"][i][0], invoiceDetail["items"][i][1],
            invoiceDetail["items"][i][2]])+"\n")
    file.close()
```

7) REFERENCES

www.w3school.com

www.geekforgeeks.com

