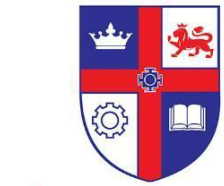
LONDON METROPOLITAN UNIVERSITY



islington college
(इस्लिङ्टन कलेज)

CS4051NI Fundamentals of Computing

60% Individual Coursework

2023/24 Spring

Student Name: Shivam Kumar Thakur

London Met ID: 23050396

College ID: NP01CP4A230301

Assignment Due Date : 7th May 2024

Assignment Submission Date: 7th May 2024

Word Count: 242

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher 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

Table Of Figures :

List of Tables :

1. Introduction

1.1 Python

Python is a programming language known for its simplicity, versatility, and readability. It was developed by Guido van Rossum and released in 1991, Python is an interpreted, object-oriented and a high-level language which is designed to be user-friendly, and its language's syntax is easy to learn and emphasizes readability. It has gained popularity as an introductory language due to its ease of use, allowing beginners to focus on understanding programming concepts rather than getting entangled down in technical details.

Python is mostly used in web development, data science, and artificial intelligence. As an open-source language, Python is benefitted from a large and active community of developers who continuously contribute libraries and functionality, making it a versatile and widely used language in the programming world.

1.2 About the Project

In this project, we were focused on building a program for Techno Property Nepal which would let the user to carry out rent and return transaction for managing land data stored in a text file. We were asked to use file handling to achieve this. We will have to read all the land data from a txt file and perform various functions like rent and return according to the txt file. We would have to make four different files to carry out different actions. The main.py file would prompt the user an interface or main menu and gather their input for rent or return. The operation.py file would carry out essential calculations and methods crucial for our program's functionality which will use algorithms to check the availability of lands, ensuring smooth transactions for our user. The write.py file would contain the code to generate an invoice in the text file. The read.py file would read the data from the txt file and would help other functions and provide other method and function with the values of

the txt file. Additionally, it will be responsible for printing invoices to the console whether it's for rent or return transaction. With this, we will be making an easy-to-use program that simplifies land management duties for Techno Property Nepal.

1.3 Goals of this project

The goal for this project was to get us familiar with the python programming language concepts and sharpen our knowledge about python syntax and exception handling which would help us to gain hands on experience in programming which would later benefit us in the future career opportunities. This project also helped to enhance our critical thinking and problem-solving skills which is beneficial in a variety of industries beyond programming.

1.4 Objective of this project

The objective of this project was to develop a land rental system for Techno Property Nepal which would automate land rental transactions based on data stored in a text file. It would use file handling of python language to read and manipulate land data in txt file. It will be easy to use and have a user-friendly interface for returning and renting land and will function perfectly when a land is rented or returned. It will be able to read and update the file that contains information about the lands available for rent. It will have exception handling throughout the program to ensure smooth functionality. It would be updating the availability status of land after transactions, generating invoice records for transactions which would contain key details like Kitta number, direction, area, rent, duration, and availability status of the land in separate txt file. After renting, it would mark the land as "Not Available" and change it to "Available" upon return.

In simple terms, the program aims to make renting land easier and faster for Techno Property Nepal. It helps automate the process of renting out land, making it more efficient and accurate.

1.5 Tools used in the Project :

IDLE

IDLE (Integrated Development and Learning Environment) is an integrated development environment (IDE) for Python. It is included with the Python programming language distribution and provides a graphical user interface (GUI) for writing, running, and debugging Python code. IDLE includes features such as syntax highlighting, code completion, and interactive Python shell, making it a convenient tool for beginners and experienced developers.

IDLE was my main tool of choice when writing Python code, so it was essential to the project. Syntax errors were greatly reduced with its help, and the provided clear and concise error messages really helped me in my project for efficient debugging. It also provided extra features that increased development productivity.

Draw.io

Draw.io is proprietary software for making diagrams and charts. The software lets you choose from an automatic layout function or create a custom layout. They have a large selection of shapes and hundreds of visual elements to make your diagram or chart one-of-a-kind. The drag-and-drop feature makes it simple to create a great looking diagram or chart. (Computer Hope, 2020)

I used draw.io's wide range of shapes and symbols to make a flowchart for the project. I was able to create a visually appealing representation of the project workflow to meet needs and made sure the project logical steps were shown clearly.

Microsoft Word

Microsoft Word is a widely used word processing software created by Microsoft. It is used for creating, editing, and formatting documents like letters, reports, and resumes. With features like spell check, formatting tools, and collaboration options, it's a versatile tool for various tasks.

2. Algorithm

An algorithm can be defined as a step-by-step procedure or a set of rules designed to solve a specific problem or perform a particular task. It is essentially a precise set of instructions that, when followed in the correct order, leads to the desired outcome.

Algorithms can be found in various fields, including mathematics, computer science, and everyday life. They are used to solve complex problems, make predictions, and automate tasks. In computer science, algorithms are particularly vital as they form the foundation of software development and programming. (Linkedin, 2023)

2.1 Algorithm for the Coursework :

Step 1: Initiate the "TECHNO PROPERTY NEPAL" Land Rental System.

Step 2: Greet user with a welcome message and show all the land details.

Step 3: Prompt the administrator to select an action:

1) Rent Lands
2) Return Lands
3) Exit.

Step 4: If the administrator chooses to Rent Lands, proceed to Step 5.

Elif Return Lands is selected, proceed to Step 12.

Elif Exit is opted for, proceed to Step 18.

Step 5: Gather necessary details from the customer, including their name, phone number and the kitta numbers of the desired lands for rental system.

Step 6: Update the availability status of the selected lands to "Not Available".

Step 7: Prompt the user asking if the customer want to rent more lands or not.

Step 8: If customer wants to rent more lands, ask for KittaNo of land which he wants to rent and proceed to Step 6.
Else proceed to Step 9.

Step 9: Generate an invoice for the rental transaction.

Step 10: Prompt the user asking if he/she want to continue using the program or not.

Step 11: If the user selects to continue using the program, proceed to Step 4
Else proceed to Step 14.

Step 12: Initiate the return process, Gather the kitta numbers of the lands the customer wishes to return.

Step 13: Update the availability status of the returned lands to "Available".

Step 14: Prompt the user asking if the customer want to return more lands or not.

Step 15: If customer wants to return more lands, proceed to Step 12.
Else proceed to Step 16.

Step 16: Generate an invoice, which may include fines if applicable, for the return transaction.

Step 17: Proceed to Step 10

Step 18: Exit the program.

3. Flowchart

Flowcharts are nothing but the graphical representation of the data or the algorithm for a better understanding of the code visually. It displays step-by-step solutions to a problem, algorithm, or process. It is a pictorial way of representing steps that are preferred by most beginner-level programmers to understand algorithms of computer science, thus it contributes to troubleshooting the issues in the algorithm. A flowchart is a picture of boxes that indicates the process flow sequentially. Since a flowchart is a pictorial representation of a process or algorithm, it’s easy to interpret and understand the process. To draw a flowchart, certain rules need to be followed which are followed by all professionals to draw a flowchart and are widely accepted all over the countries. (GeeksforGeeks, 2023)

In simple words, A flowchart is a type of diagram that illustrates the steps of a process or task. It uses shapes and arrows to represent each step and the sequence in which they occur. It's like a visual roadmap that helps us understand and follow a series of actions or decisions to achieve a specific goal.

3.1 Flowchart of the coursework :

Figure 1 : Flowchart

4. Pseudocode

Pseudocode is a detailed yet readable description of what a computer program or algorithm should do. It is written in a formal yet readable style that uses a natural syntax and formatting so it can be easily understood by programmers and others involved in the development process. Pseudocode is not a programming language and cannot be compiled into an executable program. Instead, it serves as a blueprint for translating the code's logic into an actual programming language.

When pseudocode is incorporated into the development process, designers, lead programmers and other key players can use the code to design, collaborate on and evaluate the logic behind the program or algorithm. Pseudocode also provides programmers with a detailed template for writing code in a specific programming language.

Because pseudocode is written in readable format, it can be inspected by a team of designers and programmers as a way to ensure that the actual programming will match design specifications. Catching errors during the pseudocode stage is less costly than catching them later in the development process. Once the pseudocode is accepted, it can be translated into the vocabulary and syntax of a programming language. In some cases, the same pseudocode might be turned into multiple programming languages.

4.1 Pseudocode for main.py :

main.py
Import necessary modules IMPORT rent FROM operation IMPORT returnLand FROM operation IMPORT rentingBill FROM write IMPORT returningBill FROM write IMPORT changeStatus FROM write IMPORT displayLandInfo FROM read   ''' Entry point for the Land Rental System. Displays welcome message, prompts user for option (Rent, Return, or Exit), and executes action. ''' FUNCTION displayWelcomeMessage():   PRINT "Welcome to Techno Property Nepal!"

```
    displayLandInfo()



FUNCTION selectOption():

    PRINT "Please select an option:"

    PRINT "1. Rent"

    PRINT "2. Return"

    PRINT "3. Exit"

    RETURN INPUT "Enter your choice: "



FUNCTION processRent():

    data = rent()

    rentingBill(data)



FUNCTION processReturn():

    returnMore = True

    WHILE returnMore:

        returningLandID = returnLand()

        returningBill(returningLandID)
```

```
        changeStatus(returningLandID, "None", 0, "Available")

        ans = INPUT "Do you want to return more land? (y/n): "

        IF ans EQUALS "n" OR ans EQUALS "no":

            returnMore = False

        END IF



FUNCTION processExit():

    PRINT "Thank you for using our Land Rental System!"

    EXIT



FUNCTION continueRunningSystem():

    WHILE True:

        choice = INPUT "Do you want to continue? (y/n): "

        IF choice EQUALS "n" OR choice EQUALS "no":

            EXIT

        ELSE IF choice EQUALS "y" OR choice EQUALS "yes":

            BREAK

        ELSE:
```

```
            PRINT "Invalid input. Please enter 'y' or 'n'."
        END IF
    END WHILE
# Main program logic
displayWelcomeMessage()
WHILE True:
    option = selectOption()
    IF option EQUALS "1" OR option EQUALS "rent":
        processRent()
    ELSE IF option EQUALS "2" OR option EQUALS "return":
        processReturn()
    ELSE IF option EQUALS "3" OR option EQUALS "exit":
        processExit()
    END IF
    continueRunningSystem()
END WHILE
```

Table 1:Pseudocode for main.py

4.2 Pseudocode for operation.py

operation.py

```
# Importing necessary modules

IMPORT getLandsData FROM read

IMPORT changeStatus FROM write

IMPORT formatData FROM read



FUNCTION rent():

    PRINT "Enter customer name: "

    READ name

    WHILE True:

        IF LENGTH(name) < 3:

            PRINT "Please enter a valid name"

        ELSE:

            BREAK

        END IF

    END WHILE
```

```
PRINT "Enter customer phone: "
READ phone
WHILE True:
    TRY:
        READ phone
        IF LENGTH(phone) < 10 AND CAST(phone) < 9000000000:
            PRINT "Enter a valid phone number"
        ELSE:
            BREAK
        END IF
    EXCEPT:
        PRINT "Please enter a valid phone number"
    END TRY
END WHILE
landsDataDict = CALL getLandsData()
rentingList = []
rentingDuration = []
```

```
    WHILE True:
        TRY:
            READ landID
            IF landID NOT IN landsDataDict:
                PRINT "Enter a valid land id."
                CONTINUE
            ELSE IF landID IN rentingList:
                PRINT "This land is already added to your invoice. Please select another one."
                CONTINUE
            ELSE IF formatData(landsDataDict[landID][4]) == "Not Available":
                PRINT "This land is already rented. Please select another one."
                CONTINUE
            ELSE:
                APPEND landID TO rentingList
        EXCEPT:
            PRINT "Enter a valid land id."


        WHILE True:
```

```
        TRY:
            READ duration
            IF duration < 1 OR duration > 360:
                PRINT "Enter months between 1 and 360 months."
                CONTINUE
            ELSE:
                APPEND duration TO rentingDuration
                BREAK
        EXCEPT:
            PRINT "Enter valid number of months."
    PRINT "Do he/she wants to rent more land(y/n): "
    READ choice
    IF choice == "yes" OR choice == "y":
        CONTINUE
    ELSE:
        BREAK
FOR i IN RANGE(LENGTH(rentingList)):
    CALL changeStatus(rentingList[i], name, rentingDuration[i], "Not Available")
```

```
    RETURN [name, phone, rentingList, rentingDuration]


FUNCTION returnLand():
    d = CALL getLandsData()


    WHILE True:
        READ landID
        IF landID NOT IN d:
            PRINT "Enter a valid land id."
            CONTINUE
        ELSE IF formatData(d[landID][4]) == "Available":
            PRINT "This land is not rented yet. Please select another one."
            CONTINUE
        ELSE:
            BREAK
    RETURN landID
```

Table 2:Pseudocode for operation.py

4.3. **Pseudocode for read.py**

read.py

```
DECLARE FUNCTION formatData(string, length=0)
    IF length EQUALS 0 THEN
        RETURN string
    ELSE IF length > LEN(string) THEN
        WHILE length NOT EQUALS LEN(string) DO
            string += " "
        END WHILE
        RETURN string
    ELSE IF length < LEN(string) THEN
        RETURN string[0:length]
    END IF
    RETURN string

DECLARE FUNCTION displayLandInfo()
    OPEN file at filePath in read mode
    READ lines from file
    PRINT header for the table
    PRINT table column headers
    FOR each line in lines DO
        REMOVE newline characters and split line into list
        PRINT formatted data for each land
    END FOR
    PRINT table footer

DECLARE FUNCTION getLandsData()
    INIT landsDataDict as empty dictionary
    OPEN file at filePath in read mode
    READ lines from file
```

```
    FOR each line in lines DO
        REPLACE newline characters with space
        SPLIT line into list by comma
        ASSIGN key as formatted data from first element of line
        INIT val as empty list
        FOR i from 1 to length of line DO
            APPEND formatted data from i-th element of line to val
        END FOR
        ADD key-value pair to landsDataDict
    END FOR
    RETURN landsDataDict

DECLARE FUNCTION showBill(invoiceName)
    OPEN invoice file named invoiceName in read mode
    READ all lines from invoice
    FOR each line in allLands DO
        IF "-- Invoice --" in line THEN
            PRINT line
        ELSE
            PRINT line
        END IF
    END FOR
```

Table 3 : Pseudocode for read.py

4.4. Pseudocode for write.py

write.py

```
DECLARE FUNCTION rentingBill(data)
DECLARE FUNCTION returningBill(landID)
DECLARE FUNCTION writeNewData(d)
DECLARE FUNCTION changeStatus(id, cust, month, status)

FUNCTION rentingBill(data):
    name = data[0]
    phone = data[1]
    rentingList = data[2]
    rentingDuration = data[3]
    current_date = GET_CURRENT_DATE()
    current_time = GET_CURRENT_TIME()
    invoiceID = GENERATE_RANDOM_ID()
    invoiceName = "Renting-Invoice-" + invoiceID + ".txt"
    file = OPEN_FILE(invoiceName, "w")
    WRITE_INVOICE_HEADER_AND_CUSTOMER_DETAILS(file, invoiceID, name,
phone, current_date, current_time)
    landsDataDict = GET_LANDS_DATA()
    grandTotal = 0
    WRITE_RENTED_LAND_DETAILS_TO_INVOICE(file, landsDataDict, rentingList,
rentingDuration)
    grandTotal = CALCULATE_GRAND_TOTAL(landsDataDict, rentingList,
rentingDuration)
    WRITE_GRAND_TOTAL_TO_INVOICE(file, grandTotal)
    CLOSE_FILE(file)
    SHOW_BILL(invoiceName)
```

```
FUNCTION returningBill(landID):
    current_date = GET_CURRENT_DATE()
    current_time = GET_CURRENT_TIME()
    invoiceID = GENERATE_RANDOM_ID()
    name = GET_CUSTOMER_NAME(landID)
    invoiceName = "Returning-Invoice-" + invoiceID + ".txt"
    file = OPEN_FILE(invoiceName, "w")
    WRITE_INVOICE_HEADER_AND_CUSTOMER_DETAILS(file, invoiceID, name,
"", current_date, current_time)
    landsDataDict = GET_LANDS_DATA()
    total, fineAmount = CALCULATE_TOTAL_AND_FINE(landsDataDict, landID)
    WRITE_RETURNED_LAND_DETAILS_TO_INVOICE(file, landsDataDict, landID,
total, fineAmount)
    WRITE_GRAND_TOTAL_TO_INVOICE(file, total)
    CLOSE_FILE(file)
    SHOW_BILL(invoiceName)

FUNCTION writeNewData(d):
    file = OPEN_FILE(fileName, "w")
    WRITE_DATA_TO_FILE(file, d)
    CLOSE_FILE(file)

FUNCTION changeStatus(id, cust, month, status):
    d = GET_LANDS_DATA()
    UPDATE_LAND_STATUS_AND_DETAILS(d, id, cust, month, status)
    WRITE_UPDATED_DATA_TO_FILE(d)
```

Table 4:Pseudocode for write.py

5. Data Structure :

A data structure is a fundamental concept in computer science that systematically organizes and stores data in a computer's memory. Think of it as a blueprint or framework that defines how data elements can be stored, accessed, and manipulated efficiently. Data structures in Python play a crucial role in various computational tasks, enabling programmers to solve complex problems and optimize algorithms. (Simplilearn, 2023)

Dictionaries are key-value pairs, where each value is associated with a unique key. This structure allows for fast retrieval of values based on their keys. Dictionaries are enclosed in curly braces, and each key-value pair is separated by a colon.

Lists are dynamic, ordered collections of elements that can hold items of various data types, including numbers, strings, and even other lists. Their defining feature is their ability to be modified, expanded, or reduced during program execution. This versatility makes lists immensely useful for scenarios where data needs to be managed flexibly. Lists are created using square brackets, and elements within them are separated by commas.

In my coursework, I choose to use combination of lists and dictionaries as the data structure for managing land information in the rental system. This hybrid approach provided a robust and efficient way to organize and manipulate data related to various parcels of land.

5.1 Data Structure for storing land data in a list from a file

In getLandsData() method,first we store all that land data in a list from the file then we iterate upon each items , we replaced the line break with an empty string and then split the area using, then on the list then for each list we make the index the key of the item of dictionary and order items as a list the value of that key. We will be appending all the key, values in a dictionary and return that dictionary.

```
def getLandsData():
    """
    Function to read land data from the file and return it as a dictionary.
    It will return dictionary containing land information.
    """

    # Initialize dictionary to store land data
    landsDataDict = {}

    # Open the file
    file = open(filePath, "r")

    # Read lines from the file
    lines = file.readlines()

    # Parse each line and add data to dictionary
    for line in lines:

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

        line = line.split(",")

        key = formatData(line[0])

        val = []

        for i in range(1, len(line)):

            val.append(formatData(line[i]))

        landsDataDict[key] = val

    return landsDataDict
```

Figure 1:Data Structure for storing land data in a list from a file

5.2 Data Structure for writing land data from dictionary to file

In writeNewData() method, To write data from dictionary to file we first I rate through each key value pair of the dictionary and start writing to the file first we write the key at the zero index and other values at first index. .

```
def writeNewData(d):
    file = open(fileName, "w")
    for id, val in d.items():
        file.write("  " + str(id) + "  ")
        for each in val:
            file.write(",  " + str(each) + "  ")
        file.write("\n")
    file.close()
```

Figure 2:Data Structure for writing land data from dictionary to file

6. Implementation of the program

The objective of this coursework was to automate Techno Property Nepal's land rental company's land management and billing procedures. This was entirely inspired by the idea of file handling in Python. It consists of four files, named main.py, right.py, operation.py, and write.py, that function as a single programme. The “**main.py**” execute the entire program, handling user interaction and calling functions from other files while other file handles their own activities like the “**read.py**” programme extracts information about the land from our text file, which includes the list of the land's details including its Kitta ID, location, direction, monthly rent, and availability. "**operation.py**” performs various operations on land properties, such as updating availability or calculating fees and all other function is dependent on it. The “**write.py**” file writes information back to the text file and generates bills for rental transactions based on calculation made by operation.py. The text file is updated and a bill with the customer's information and the detail of the rented property is generated each time a rental transaction takes place.

When the program is run by running the main.py file, the user is greeted with a welcome message and given the option to rent, return, or exit. If the user chooses to rent land, we will show the details of the land that is currently in our system, determine whether it is available, and then carry out renting transactions. If the user chooses to return land, we will provide the relevant Kitta numbers and ask the user to return the land, complete return transactions, and generate a bill invoice for that returned property. Additionally, if the land is rented for a longer period of time than the specified period, a 10% fine will be applied, and the status of the land will be updated later. If the user chooses to select option three, which is exit, the program running in the idle will be killed stop executing the main.py files.

6.1 Renting of the land :

land_info.txt file before renting :

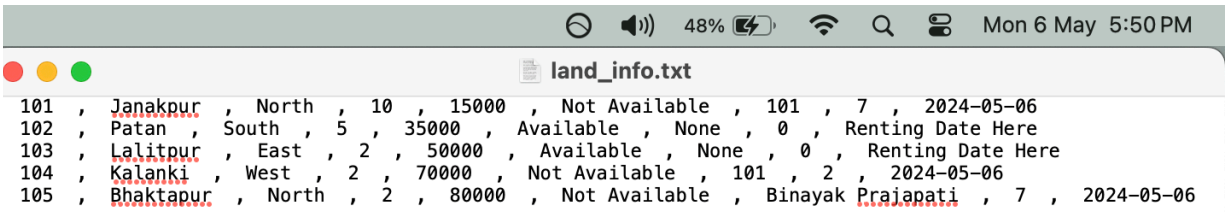


Figure 1:land_info.txt file before renting

File structure before renting a land :

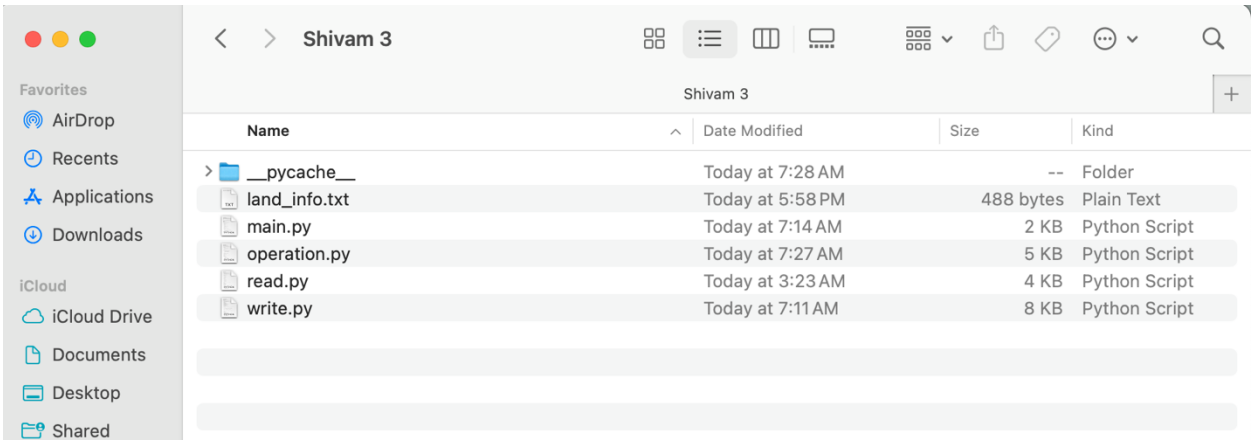


Figure 2: File structure before renting a land

Complete renting process :

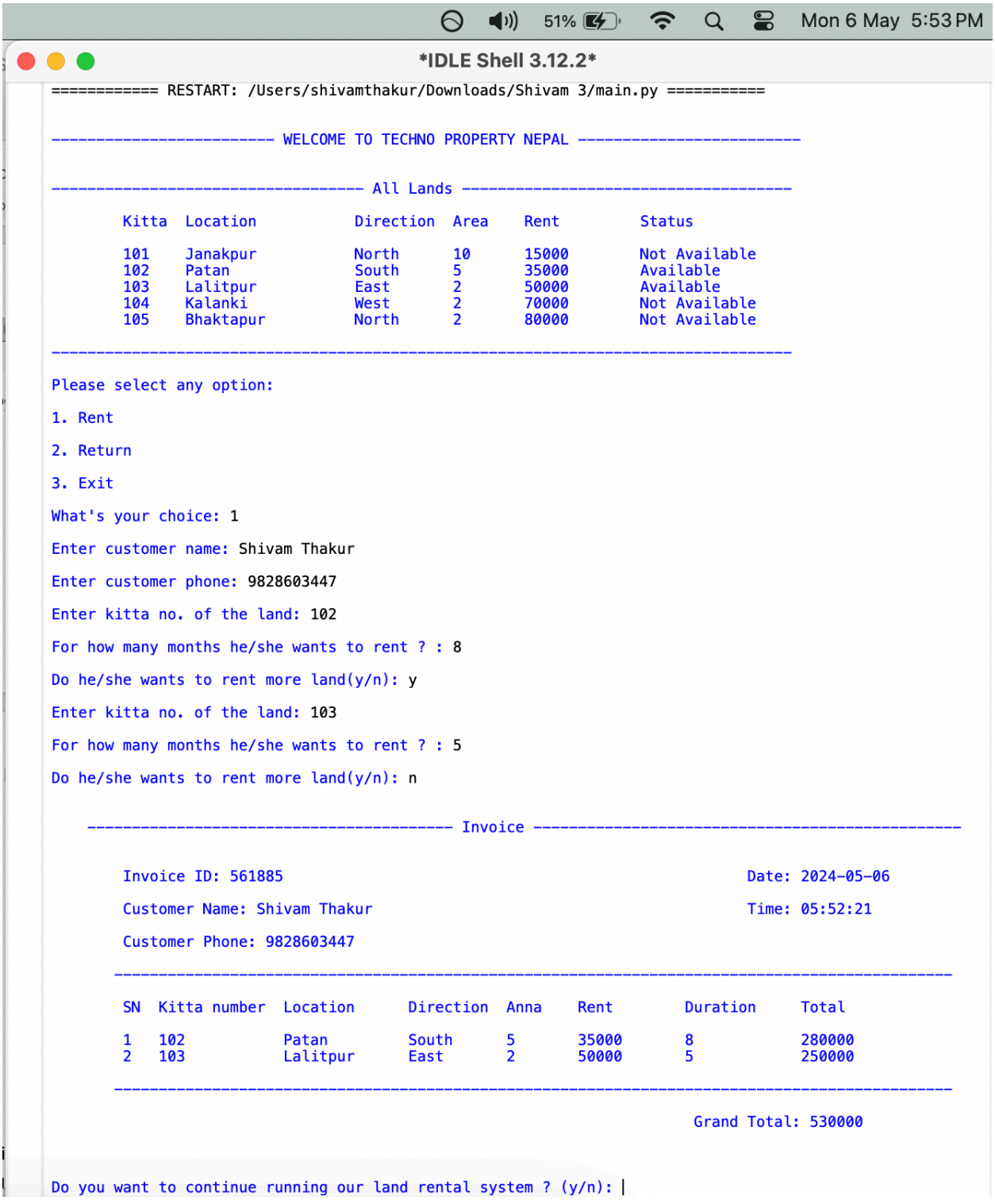


Figure 3:Complete Renting Process Screenshot 1

File Structure after renting the land and generating a rent invoice :

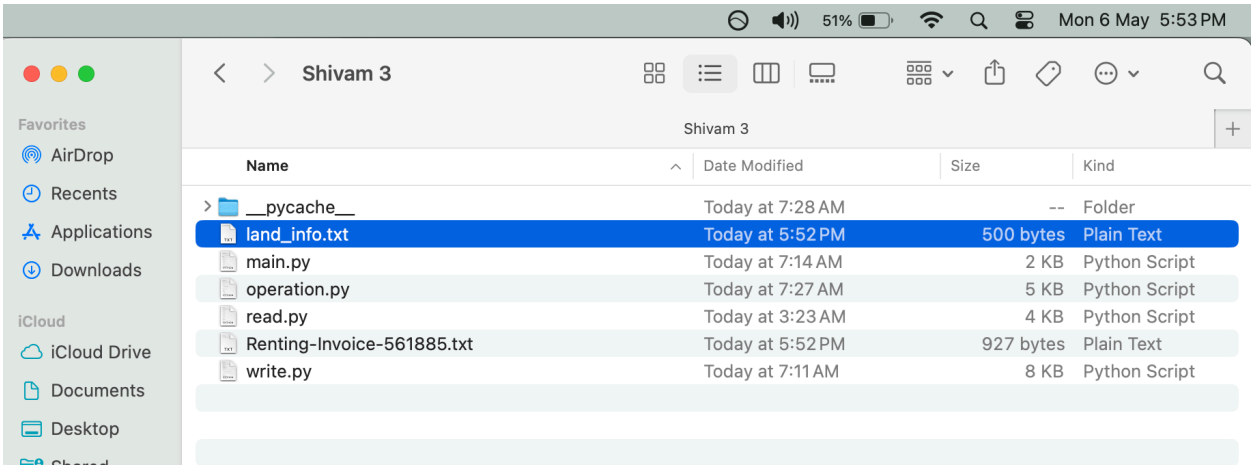


Figure 4:File Structure after renting the land

Opening the renting invoice :

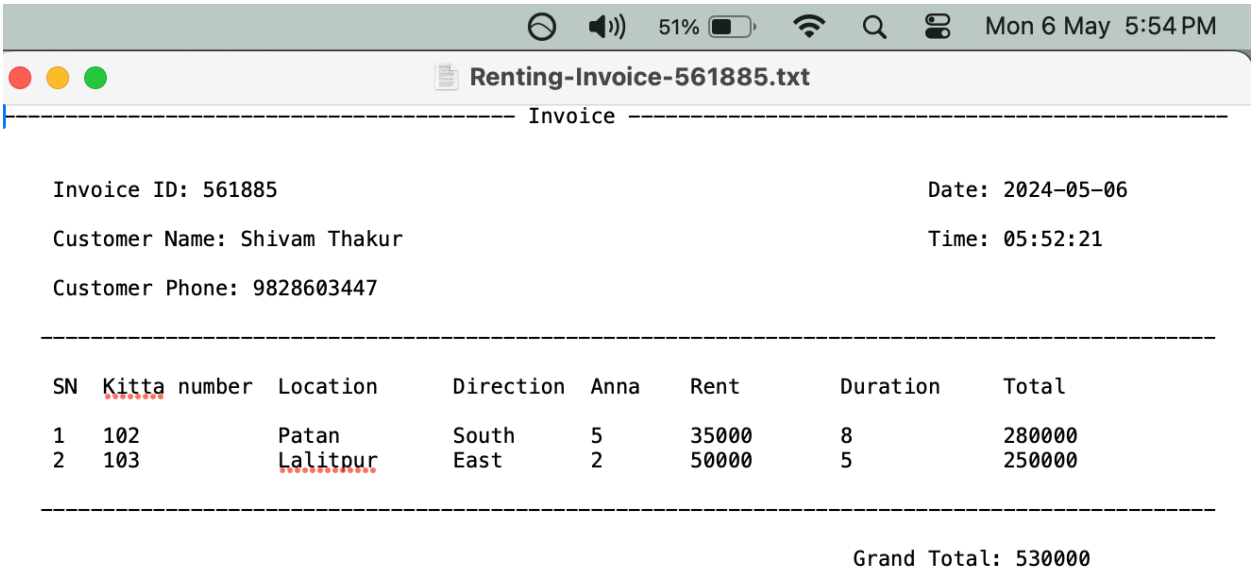


Figure 5:Renting invoice Screenshot 1

land_info.txt file after renting the land :

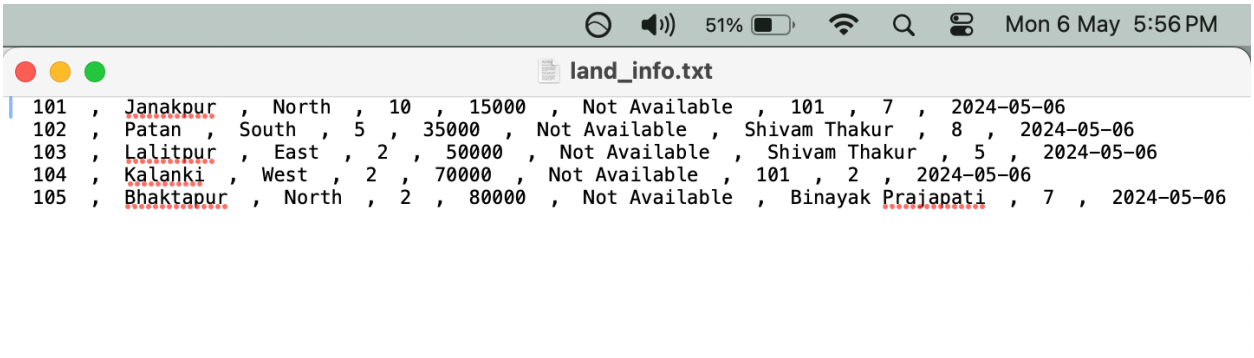


Figure 6:land_info.txt file after renting the land

6.2 Returning of land :

land_info.txt before returning land :

Figure 7:land_info.txt before returning land

Folder structure before returning the land :

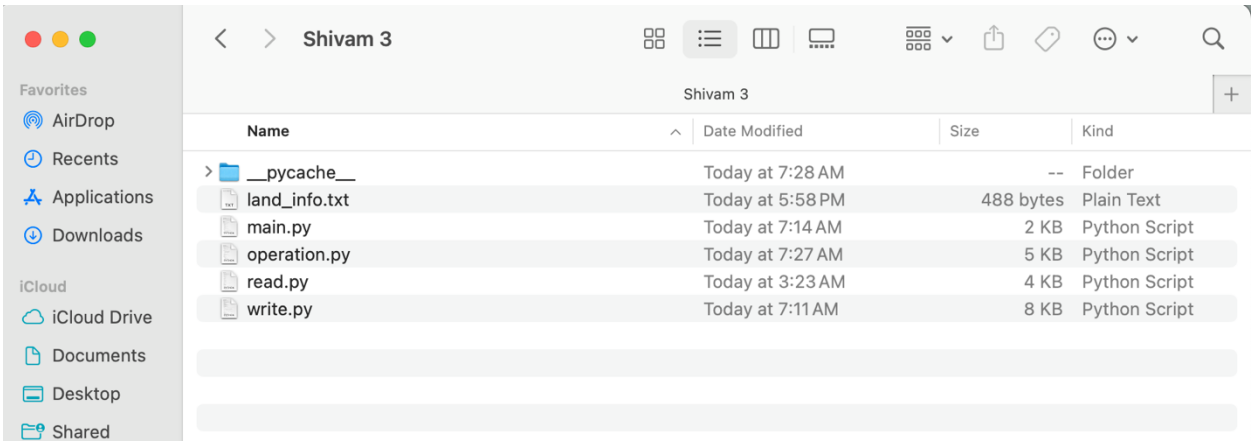


Figure 8:Folder structure before returning the land

Complete Returning Process :

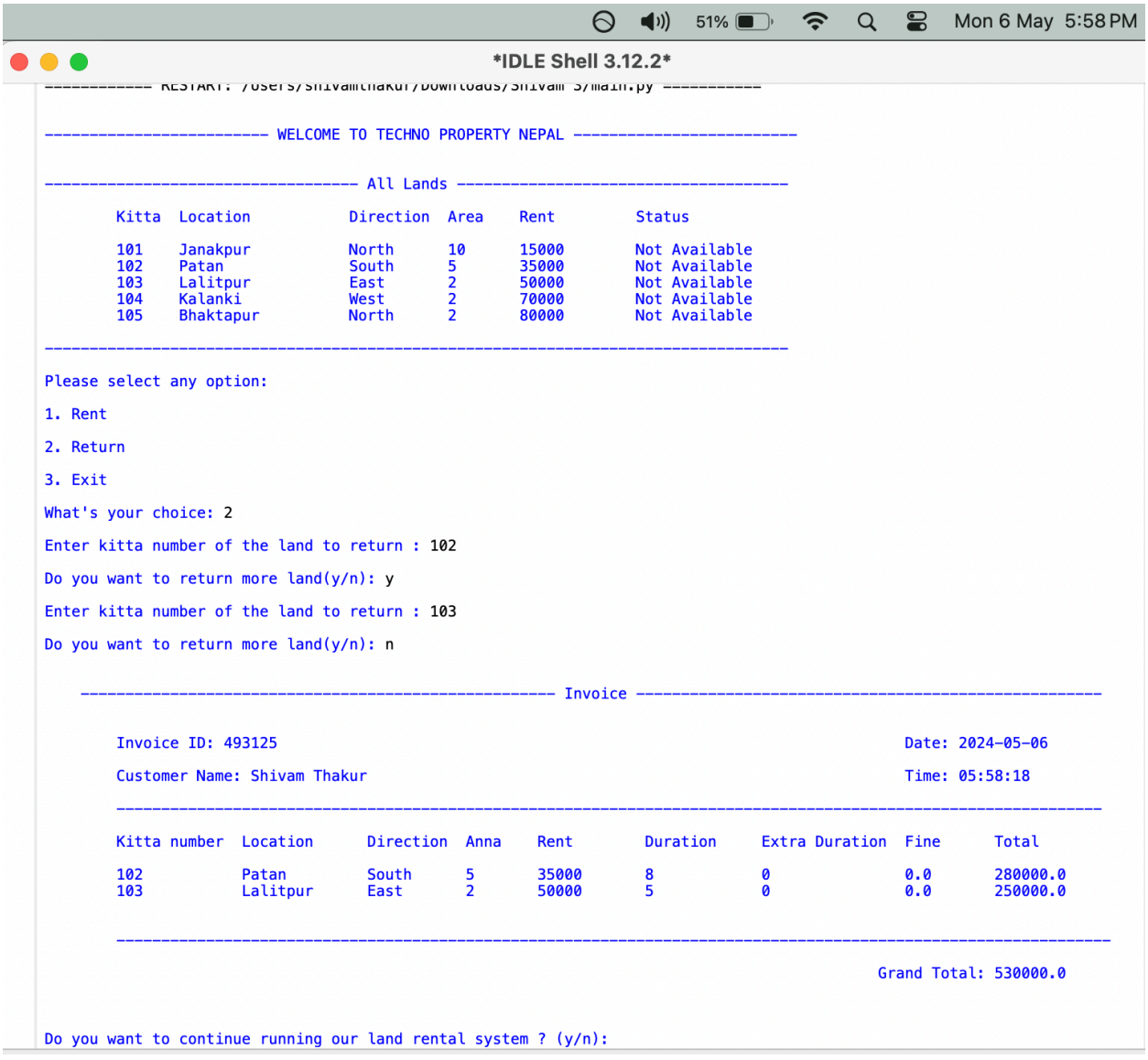


Figure 9 : Returning Complete Process

Folder Structure after returning the land :

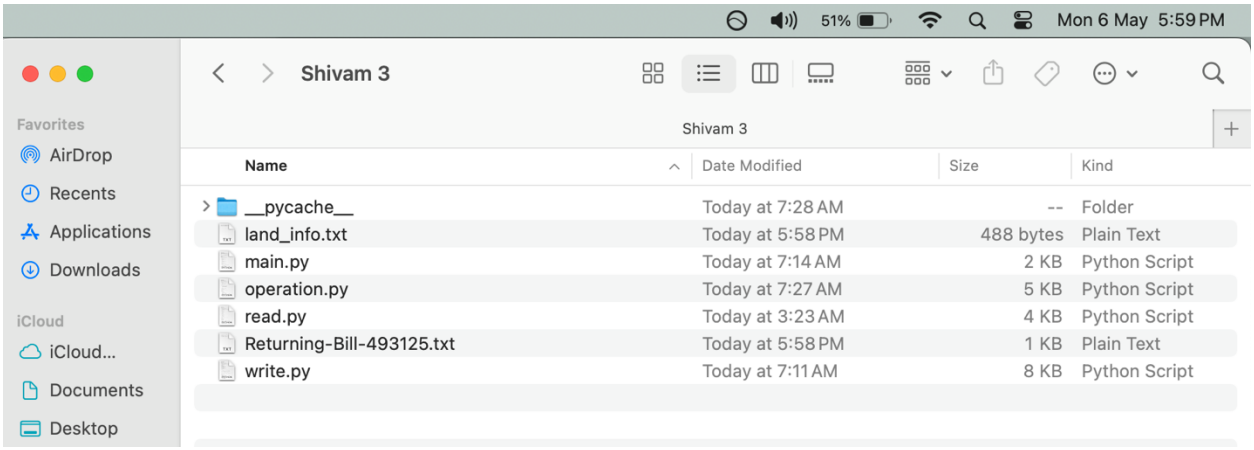


Figure 10:Folder structure after returning the lands

Opening the returning invoice :

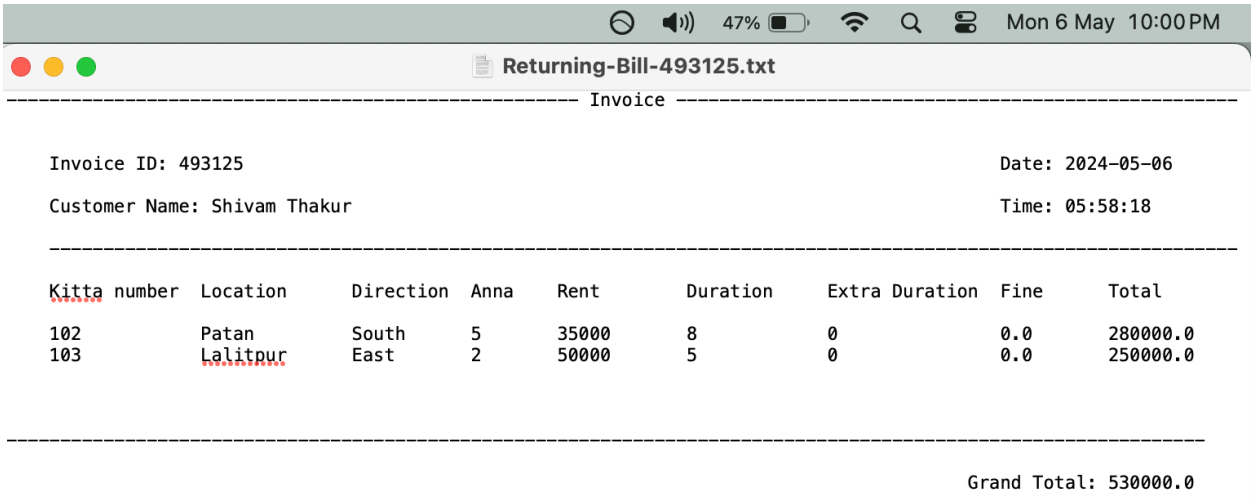


Figure 11:Return Invoice

Land_info.txt after returning land :

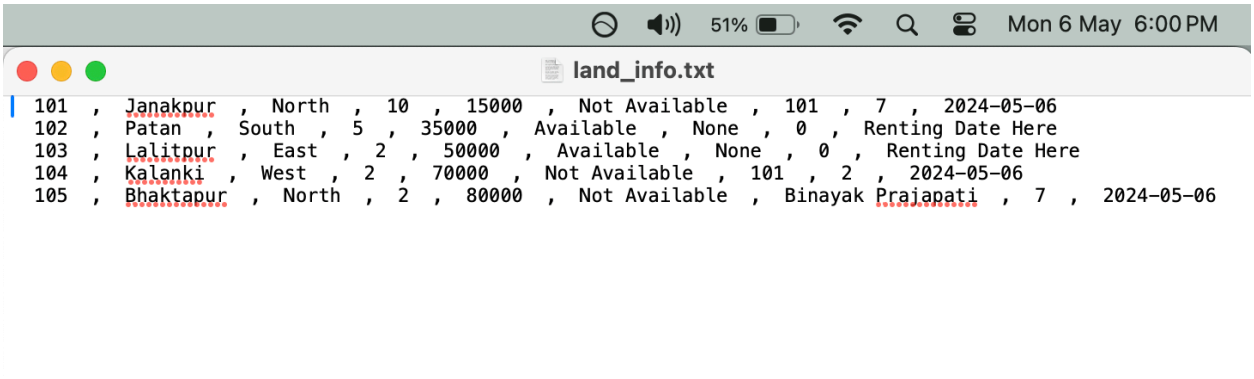


Figure 12:Land_info.txt after returning land

Termination of the code :

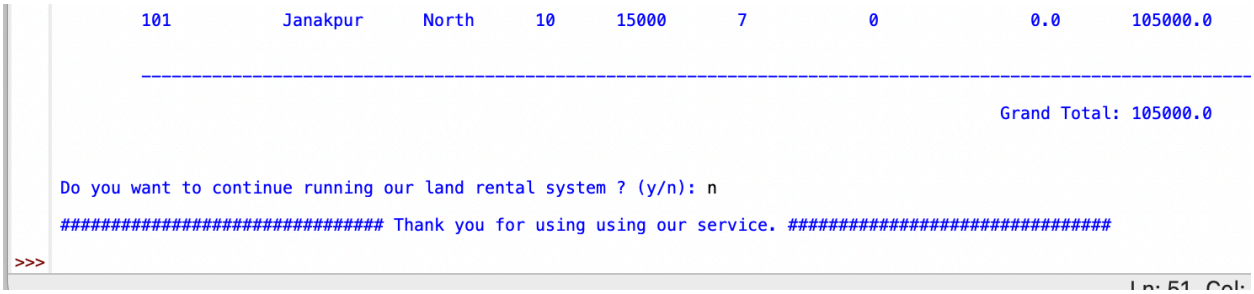


Figure 13: Termination of Program

7. Testing

7.1 Test 1 :

Showing implementation for try except in program.

Test No	1
Objective	Testing out working of Exception Handling.
Action Taken	Step 1: Executed the main.py file. Step 2 : Provided suitable values for all inputs except for the Kitta Number: • Choice = Rent (1) ; • Customer Name = “Shivam Thakur” ; • Phone Number = 9828603447 ; • **Kitta Number = #1 , n ;**
Expected Result	When entering a value that is not an integer, the program must not crash.
Actual Result	The program operated smoothly without encountering any crashes, persistently requesting input until a valid value was provided.
Conclusion	The testing passed successfully.

Table 5: Test 01

Implementation of Try & Except code block :

```
try:
    landID = input("\nEnter kitta no. of the land: ")
    if landID not in landsDataDict:
        print("\nEnter a valid land id.")
        continue
    elif landID in rentingList:
        print(
            "\nThis land is already added to your invoice. Please select another one."
        )
        continue
    elif formatData(landsDataDict[landID][4]) == "Not Available":
        print("\nThis land is already rented. Please select another one.")
        continue
    else:
        rentingList.append(landID)
except:
    print("\nEnter a valid land id.")
```

Figure 14:Implementation of Try & Except code block

Putting Exception Handling to the Test :

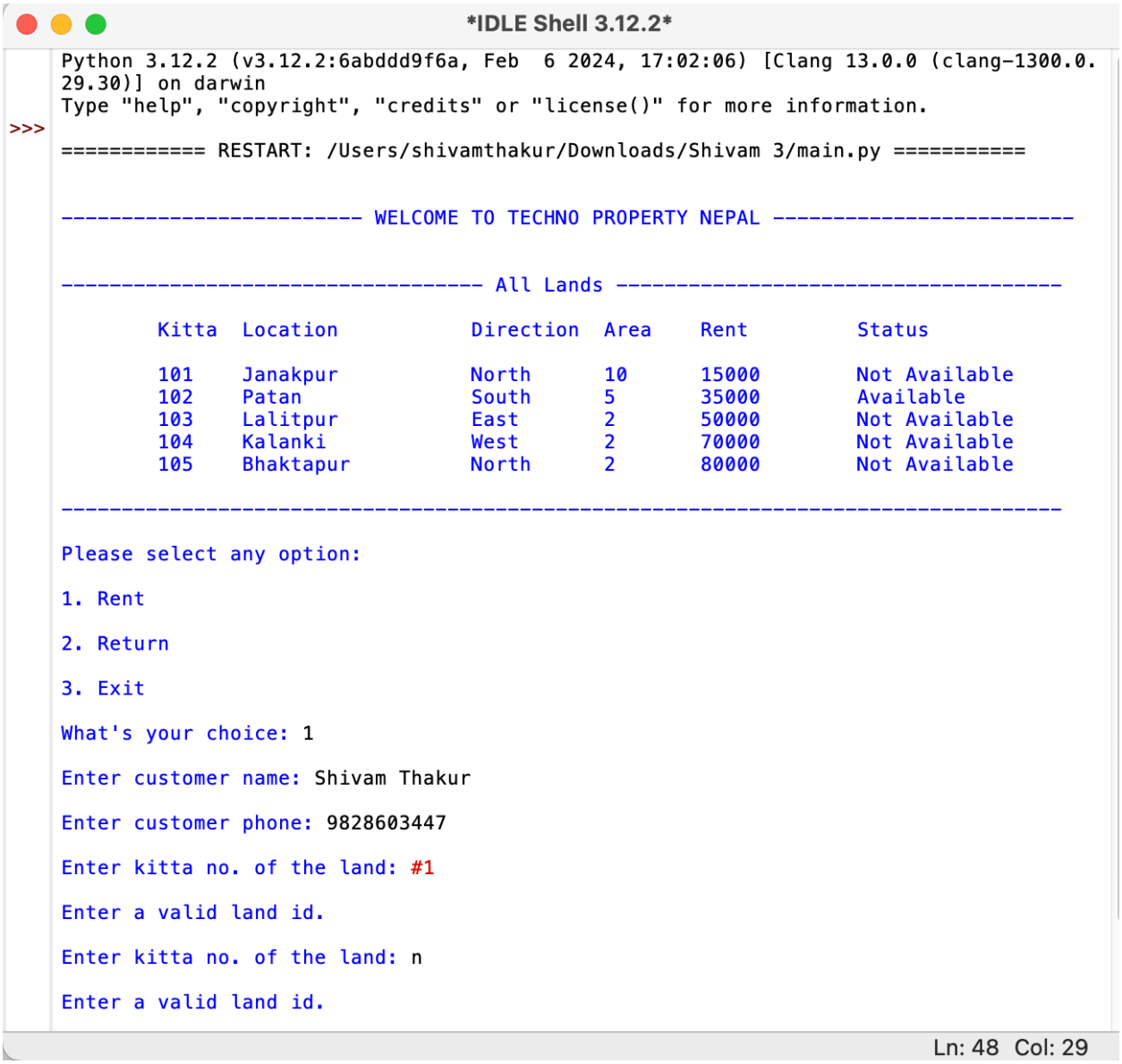


Figure 15:Testing 1 Screenshot 1

7.2 Test 2 :

Trying to enter non-existent or negative values while choosing the rent or return option and checking the outcome.

Test No	1
Objective	Entering non-existent or negative values while choosing rent or return option.
Action Taken	Step 1: Executed the main.py file. Step 2 : Provided negative and non-existent value as an option.
Expected Result	When entering a value that is negative and non-existent as an option, the program must display a error message saying that the input was not valid and continue asking user for a valid input.
Actual Result	The program displayed a message saying the input was not valid and continuously asked user for a valid input up until a valid input was provided.
Conclusion	The testing passed successfully.

Table 6:Test 02

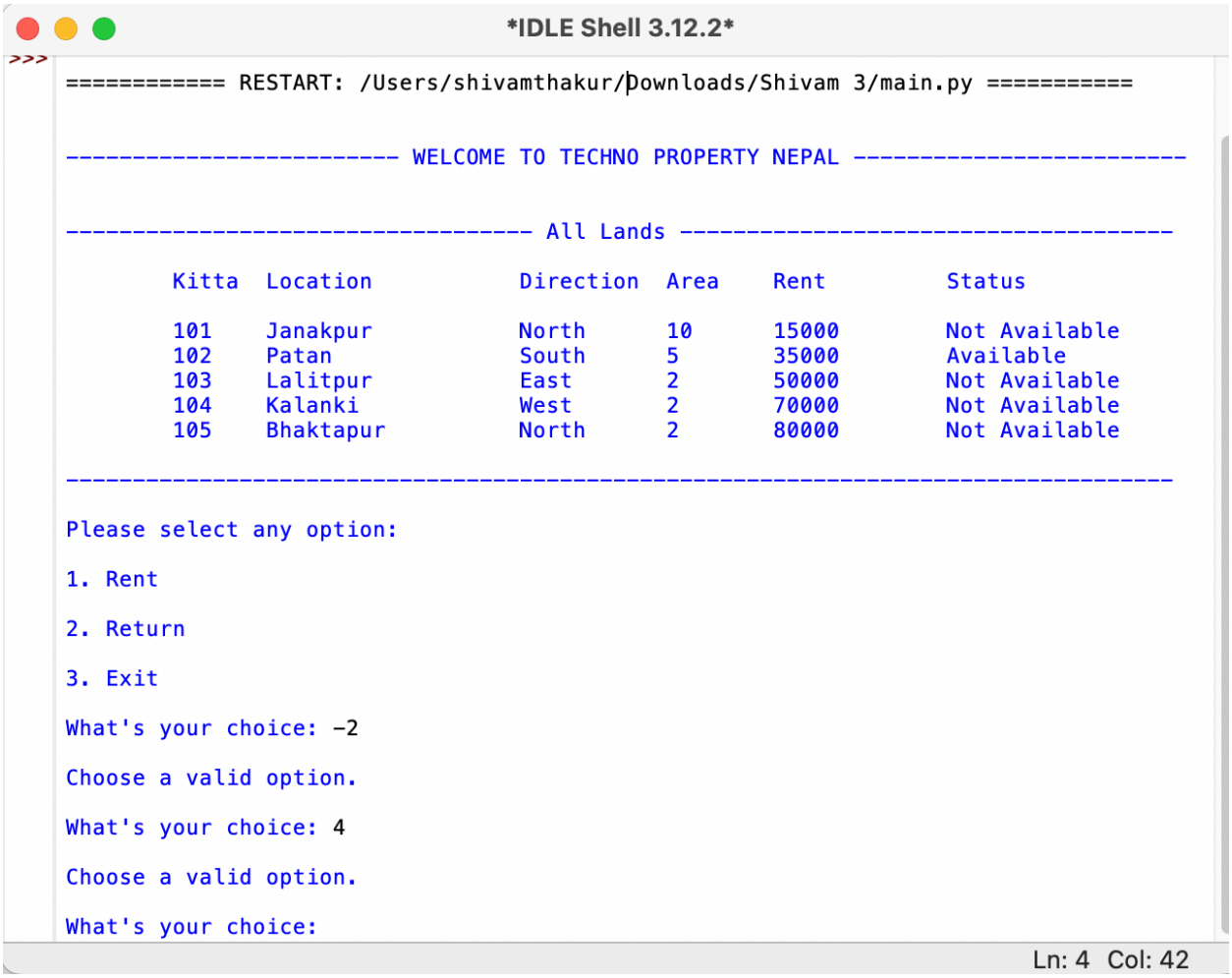


Figure 16:Testing 2 Screenshot 1

7.3 Test 3

Renting out multiple lands.

Test No	1
Objective	Renting out multiple lands and generating an invoice for the rented property.
Action Taken	Step 1: Executed the main.py file. Step 2 : Provided personal details for renting out lands. • Choice = Rent (1) ; • Customer Name = “Shivam Thakur” ; • Phone Number = 9828603447 ; Step 3 : Provided kitta number of first land and specified the lease duration for first land. • Kitta Number = 101; • Duration = 6; Step 4 : Choosed to rent more land. (y) Step 5 : Provided kitta number of second land and specified the lease duration for second land. • Kitta Number = 104; • Duration = 3; Step 6 : Choosed to not rent any more lands. (n)
Expected Result	When the land is rented out, the program should update the availability status of the land to "Not Available" and generate a renting invoice, saving it as a text file while also printing it to the console.
Actual Result	The program produced a renting invoice for the rented property, p rinted it on the console, and changed the land's availability status.
Conclusion	The testing passed successfully.

Table 7:Test 03

txt file with land information before renting land :

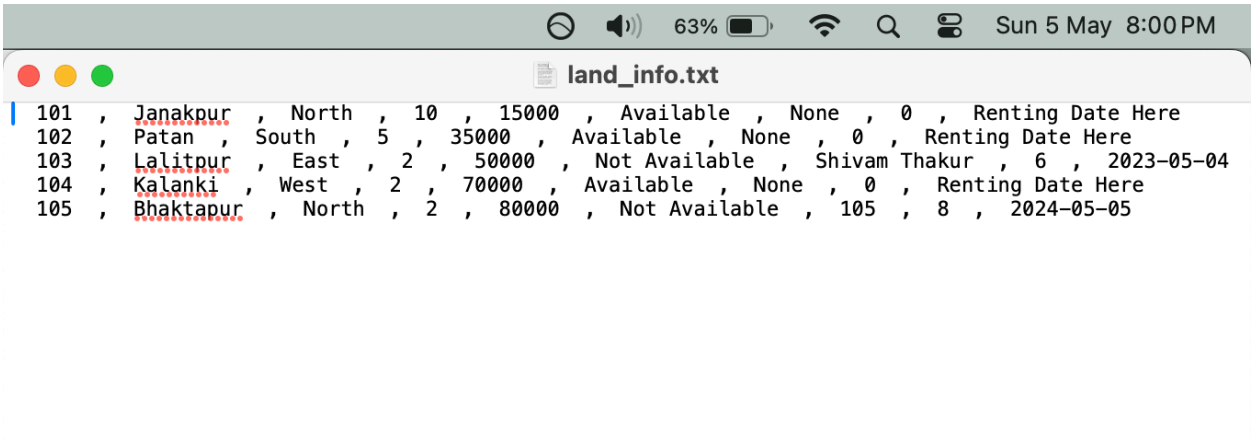


Figure 17:Test 3 Screenshot 1

txt file with land information after renting land :

Figure 18:Test 3 Screenshot 2

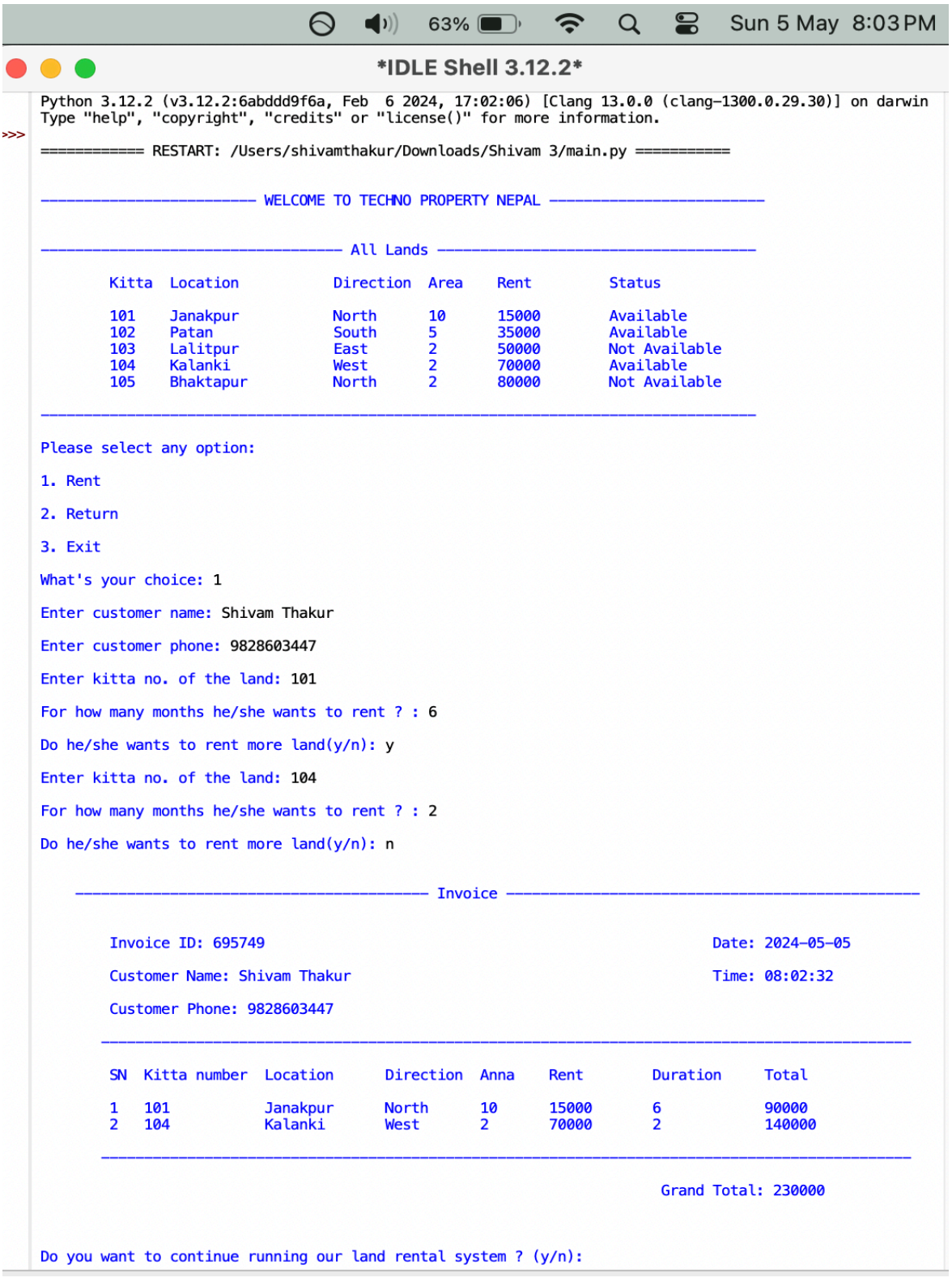


Figure 19:Test 3 Screenshot 3

Bill Invoice for rented property :

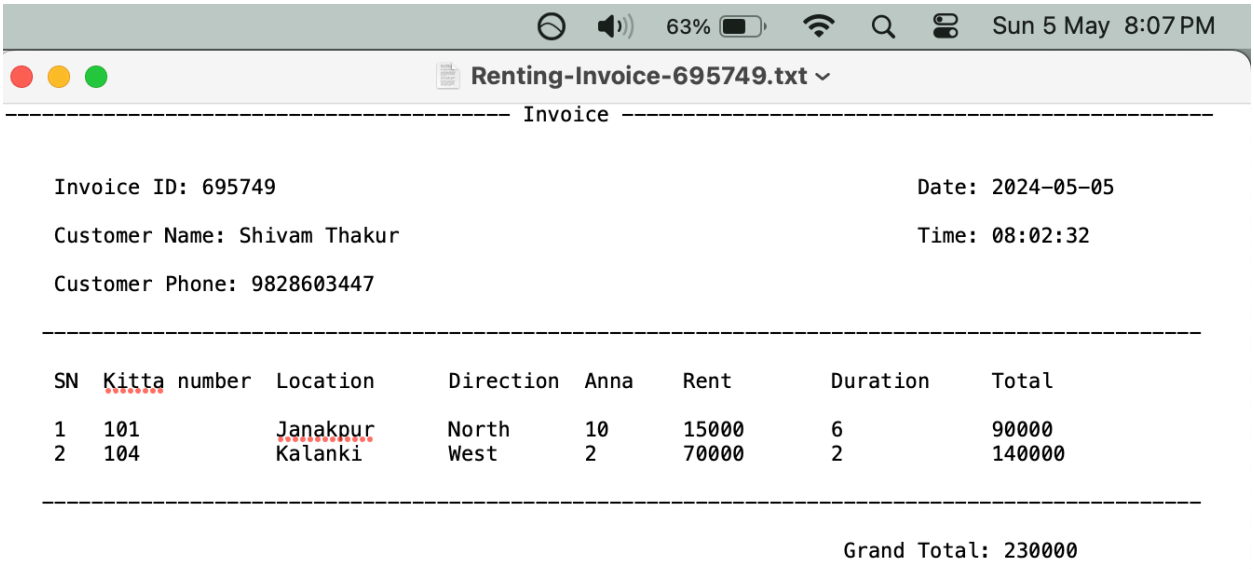


Figure 20:Test 3 Screenshot 4

Renting Invoice generation in the folder structure :

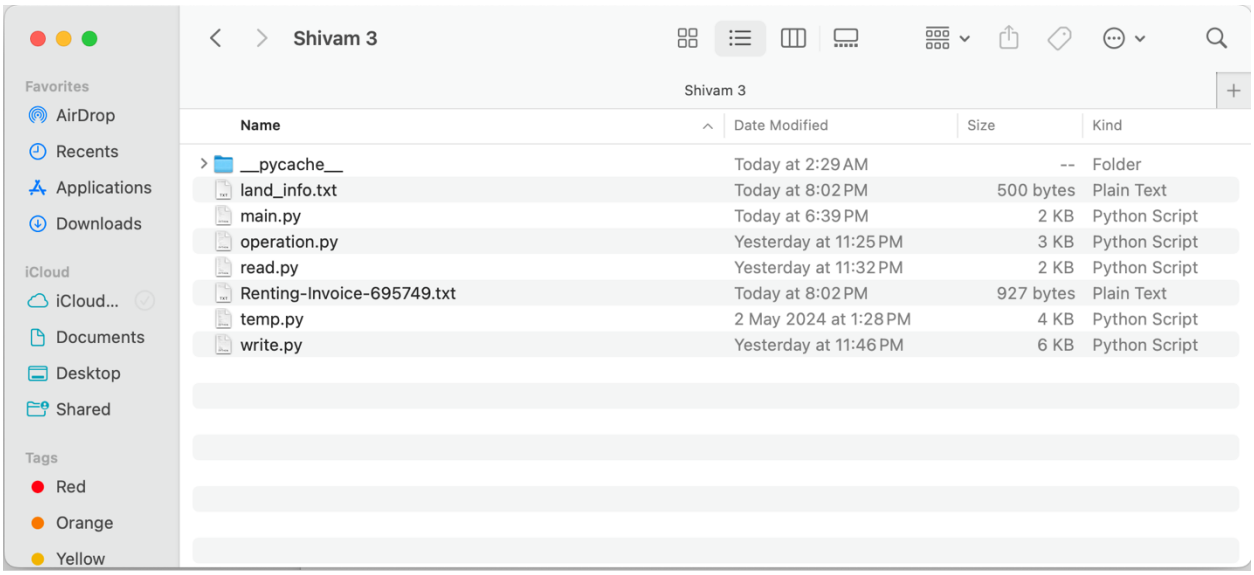


Figure 21:Test 3 Screenshot 5

7.4 Test 4 :

Renting out multiple lands.

Test No	1
Objective	Returning out multiple lands and generating an invoice for the rented property.
Action Taken	Step 1: Executed the main.py file. Step 2 : Choose the return option to return lands. Step 3 : Provided 101 as kitta number for first land to return. Step 4 : Choose to return more land. (y) Step 5 : Provided 104 as kitta number for second land to return. Step 6 : Choosed not to rent any more lands. (n)
Expected Result	When the land is returned, the program should update the availability status of the land to " Available" and generate a returning invoice, saving it as a text file while also printing it to the console.
Actual Result	The program produced a returning invoice for the returned property, printed it onto the console, and changed the land's availability status.
Conclusion	The testing passed successfully.

Table 8: Test 04

txt file with land information before renting land :

Figure 22:Testing 4 Screenshot 1

txt file with land information after renting land :

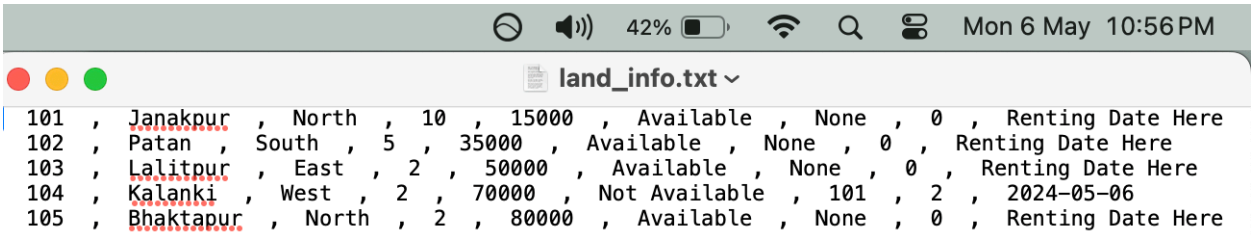


Figure 23:Testing 4 Screenshot 2

Bill Invoice for rented property :

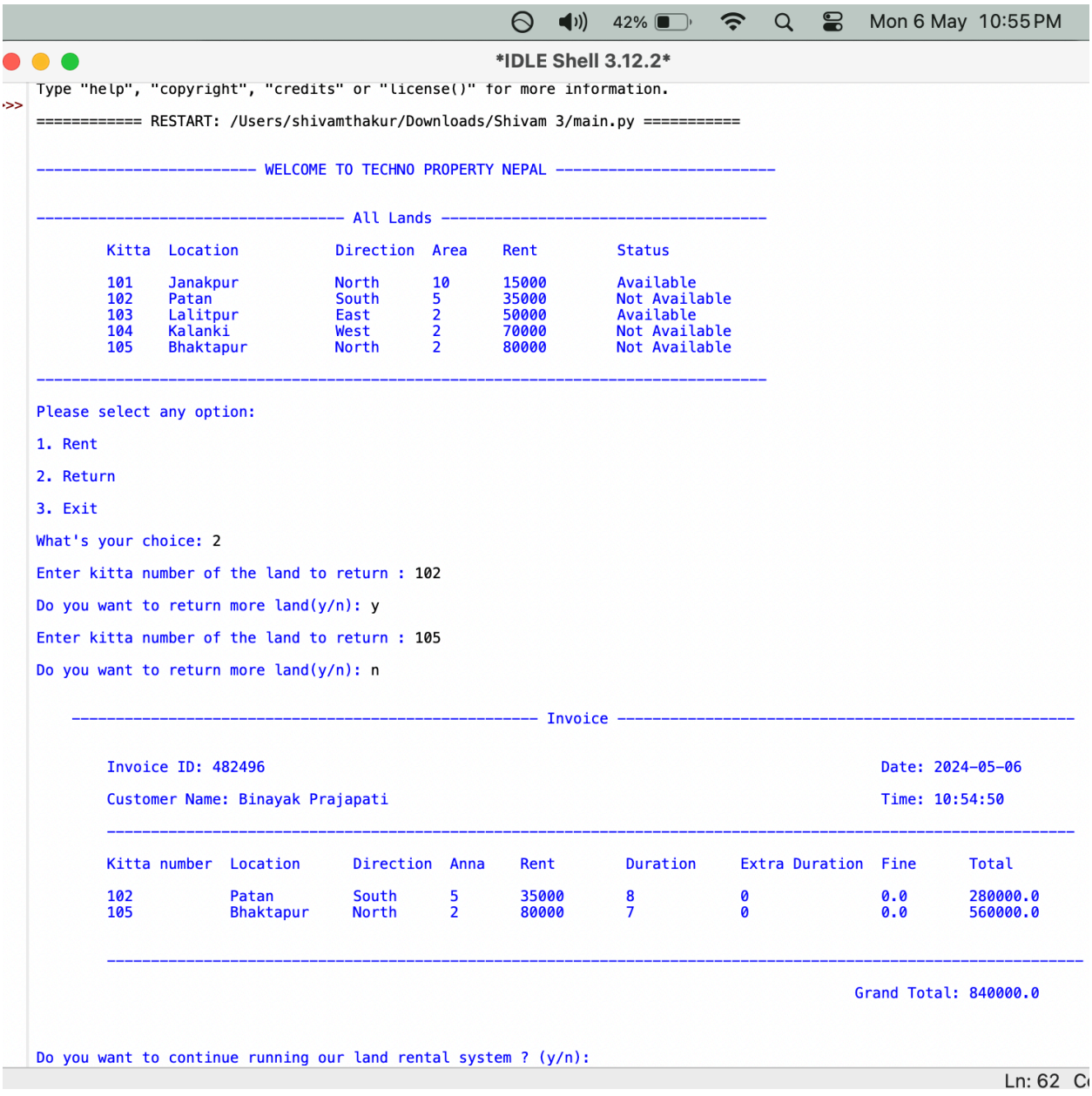


Figure 24:Testing 4 Screenshot 3

Folder Structure Before Invoice Generation :

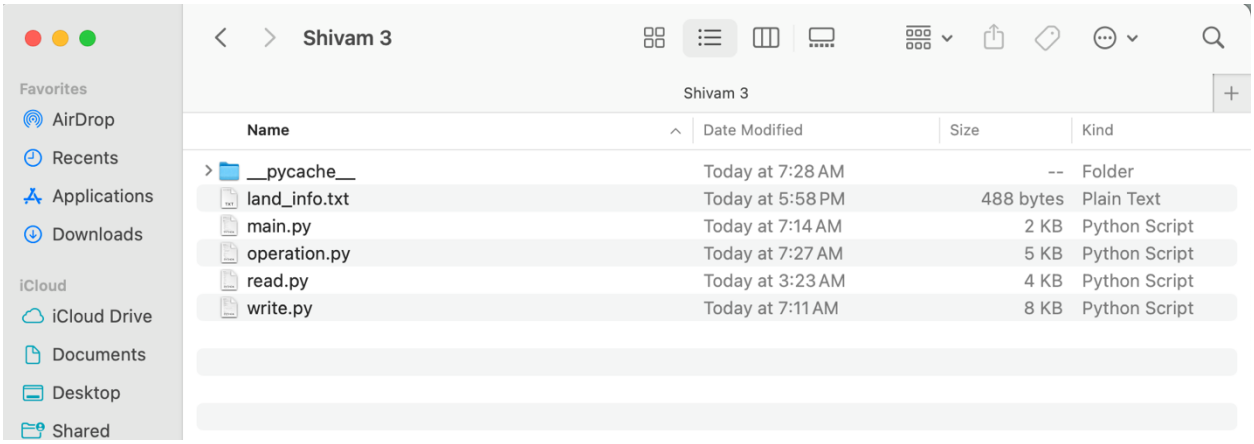


Figure 25:Testing 4 Screenshot 4

Renting Invoice generation in the folder structure :

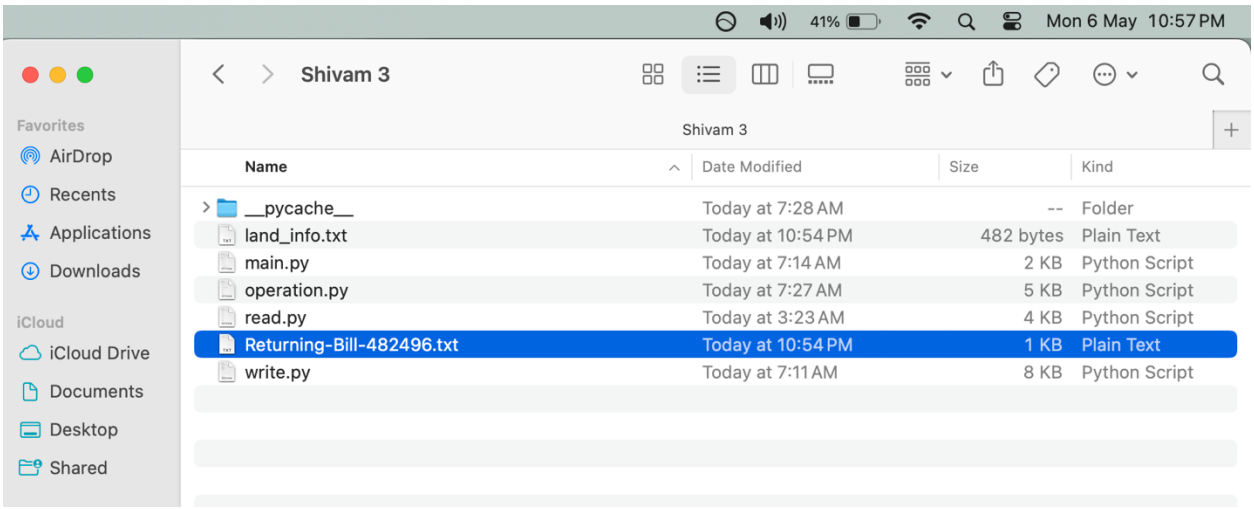


Figure 26:Testing 4 Screenshot 5

Checking Return Bill :

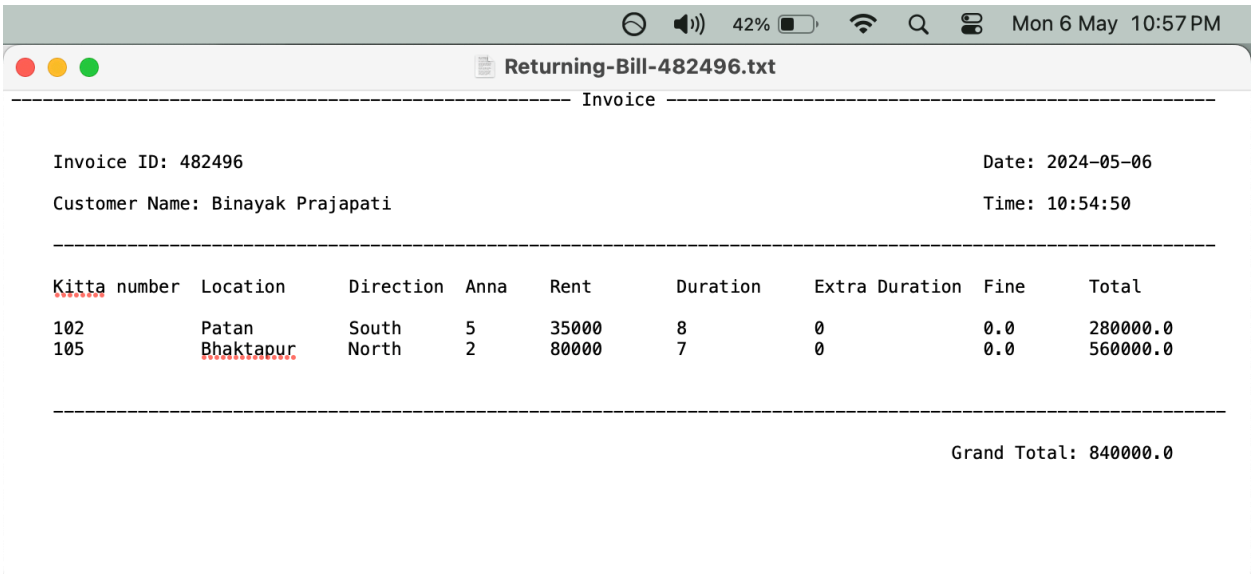


Figure 27:Testing 4 Screenshot 6

7.5 Test 5 :

Showing the changes to the availability status of the land after returning and renting of land

Test No	5
Objective	Showing the changes of the availability status of land after renting and return of the land.
Action Taken	Step 1: Executed the main.py file. Step 2 : Choose Rent option Step 3 : Rented land with ‘102’ kitta number Step 4 : Choose to not continue rent any more land Step 5 : Choose to continue using the program. Step 6 : Choose Return option. Step 7 : Returned land with ‘105’ kitta number.
Expected Result	When land is rented, its availability status is changed to "Not Available." On the other hand, the land's status will be changed to "Available" when it is returned.
Actual Result	The land.txt file was updated, and the availability of land was changed to “Not Available”. after renting land and “Available” after returning.
Conclusion	The testing passed successfully.

Table 5 : Test 05

Land.txt before renting land :

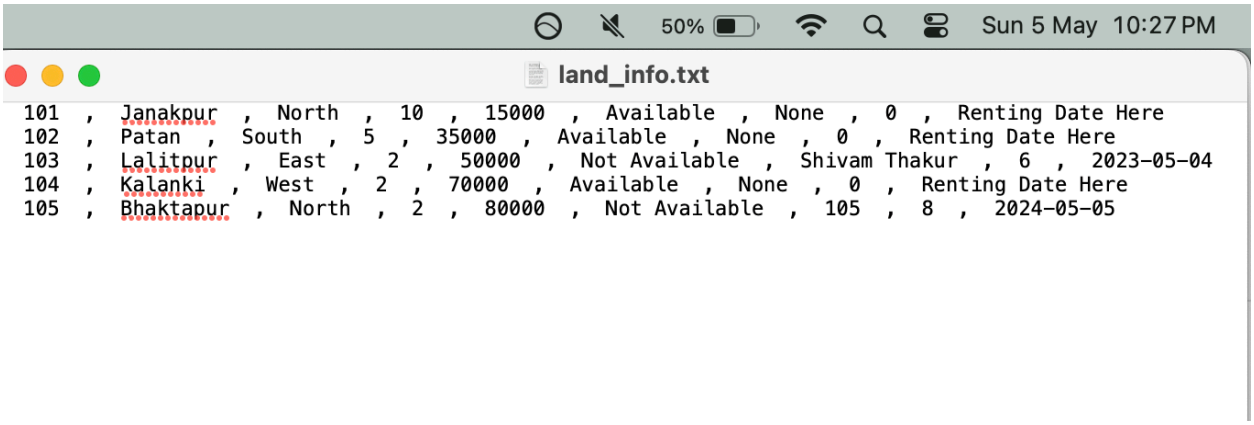


Figure 28:Testing 5 Screenshot 1

Land.txt after renting land :

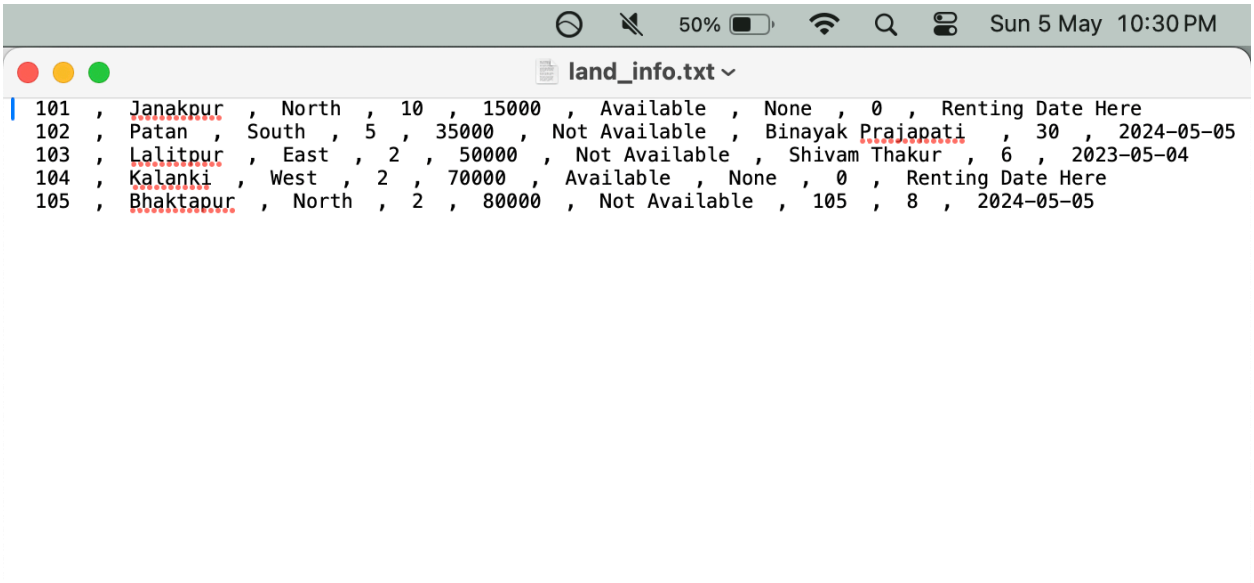


Figure 29:Testing 5 Screenshot 2

Renting Process :

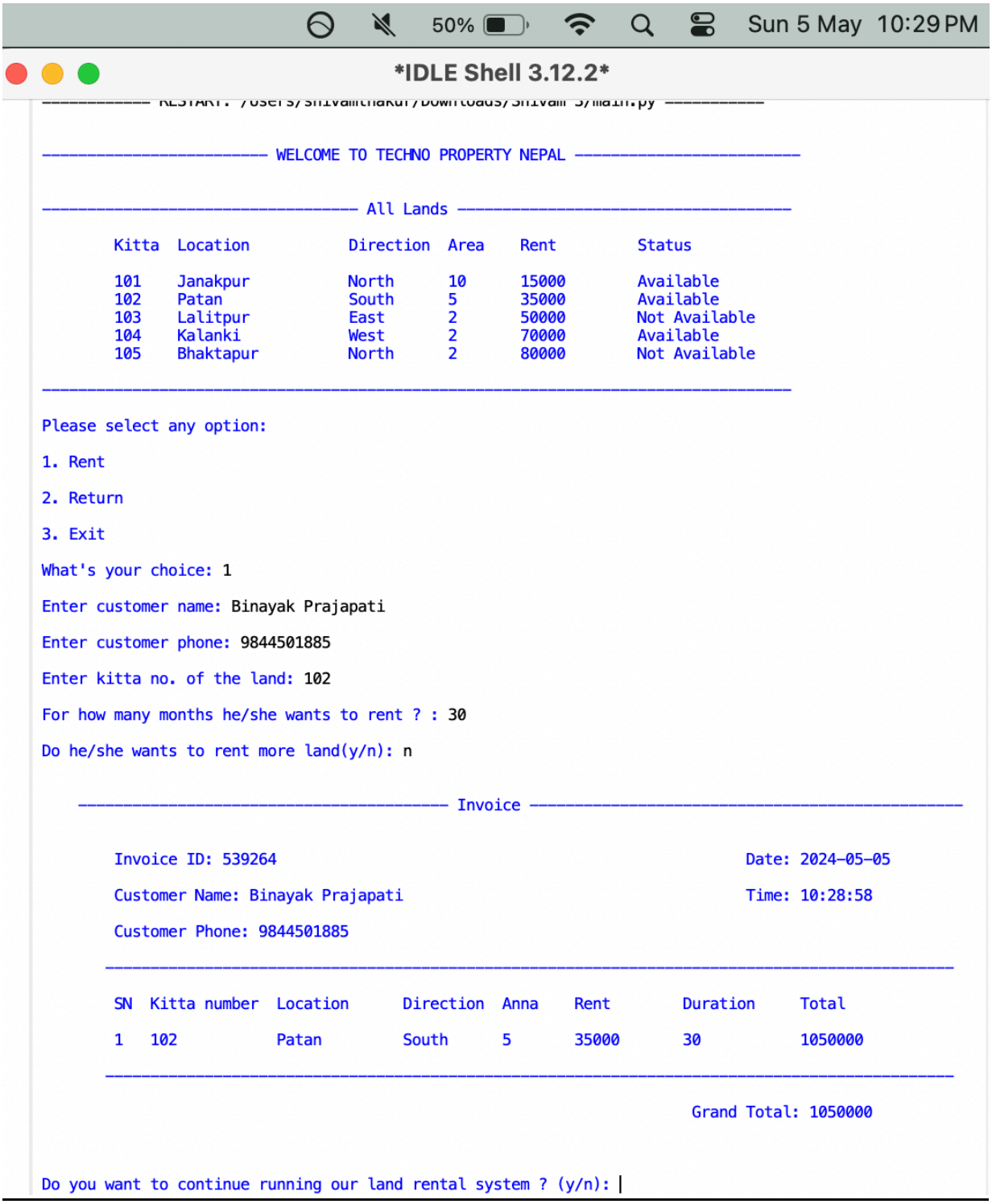


Figure 30:Testing 5 Screenshot 3

land.txt before returning land :

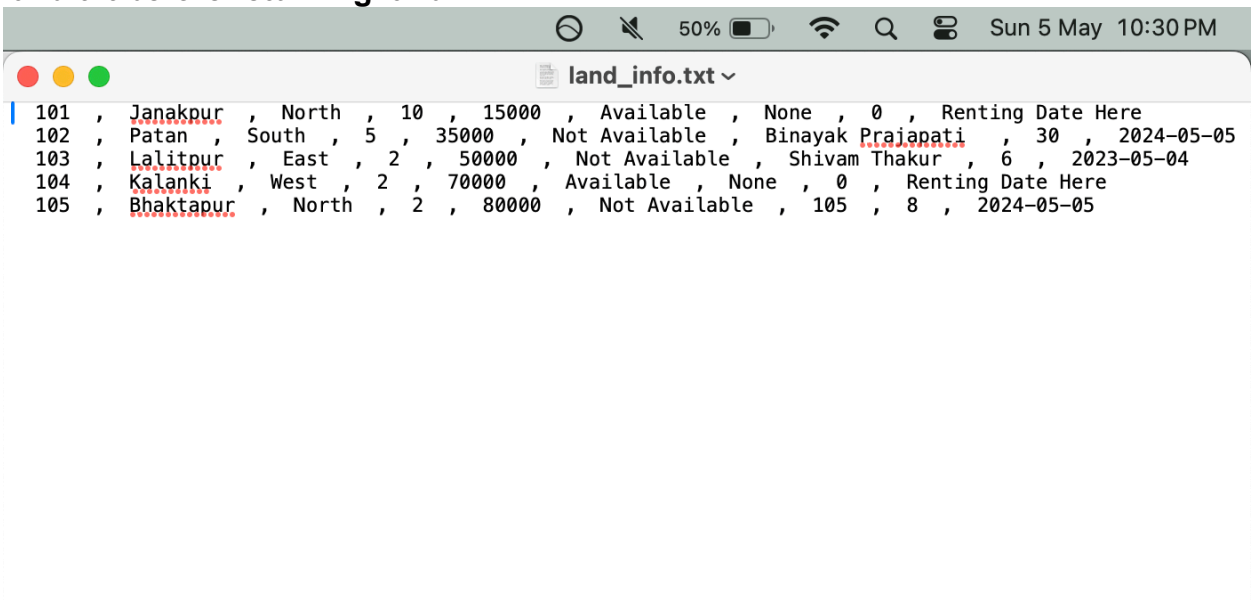


Figure 31:Testing 5 Screenshot 4

land.txt after returning land :

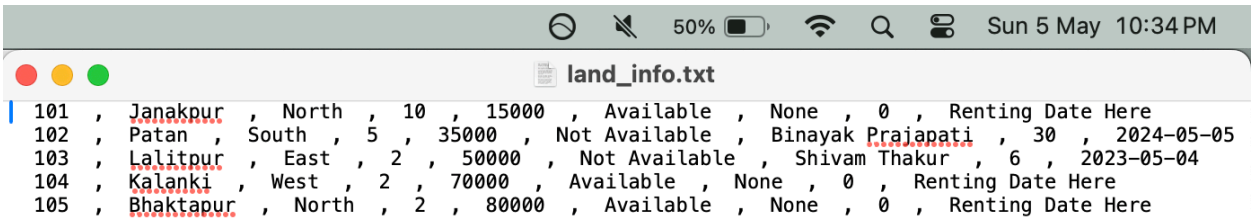


Figure 32:Testing 5 Screenshot 5

Returning Land Process :

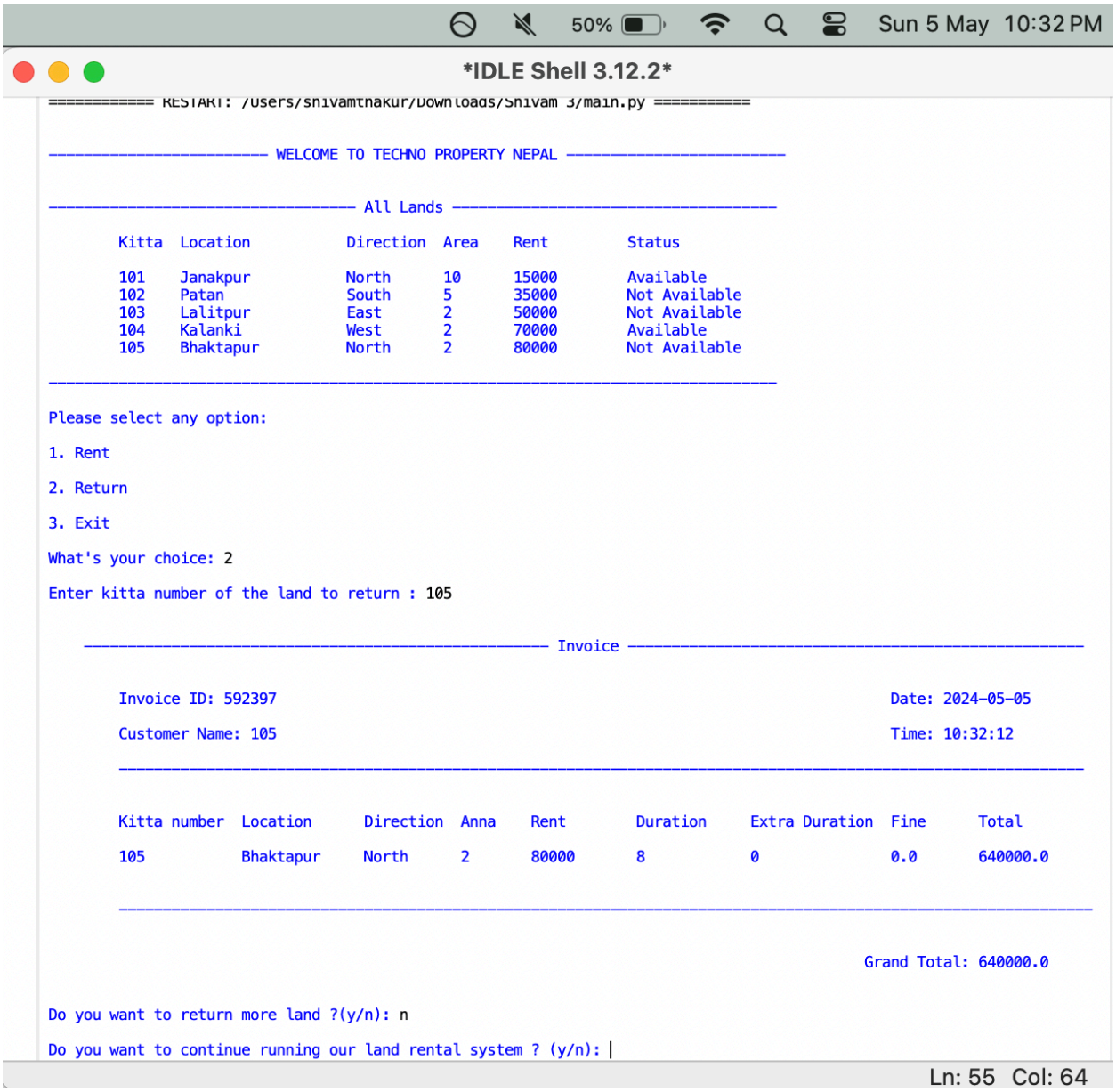


Figure 33:Testing 5 Screenshot 6

8. Conclusion

In this coursework, we were centred with developing a land rental system for Techno Property Nepal, a land rental business that specialises in seamless digital transactions for land leasing and renting. We did this by using the Python file handling concept to read land details from a text file which contain all the land details like Kitta number, direction, area, rent, duration, and availability status of the land.

Through this coursework, I utilized python programming language and its file handling concept to develop a program capable of managing land records for renting or leasing. This helped me to gain hand and experience to manipulate data effectively within files and implement various functions to facilitate different operations and actions. In this way, I boost my understanding of core python concept such as loops data structure, function methods which you gave me hands on experience for future real-life challenges and career development and also this project sharpened my problem-solving skills and algorithmic thinking.

I have encountered various difficulties while handling at the result came out as I expected it to be but however, after hours of debugging and understanding I was able to figure out the answers some of which I solved on my own and others of which I really appreciate my tutor helped me with.

I really appreciate our teacher **Mr. Bijay Gautam** and **Mr. Hrishav Tandukar** for helping me and guiding me in this course work and also helping me debug some of my problem in my coding section which I would be thankful for my entire life.

My confidence was greatly increased upon finishing the project, especially when it was time implementing what I had learned in the college about file handling and functions. This project offered valuable practical experience, illustrating how the file handling concept can be used in real-world situations .

9. Bibliography

Linkedin, 2023. *What is an Algorithm?* | *Definition, Classification & Examples.* [Online]
Available at: https://www.linkedin.com/pulse/what-algorithm-definition-classification-examples-anshul-pal-j4cuf/
[Accessed 4 5 2024].

GeeksforGeeks, 2023. *What is a Flowchart and its Types? - GeeksforGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/what-is-a-flowchart-and-its-types/
[Accessed 4 5 2024].

Computer Hope, 2020. *What is draw.io ?.* [Online]
Available at: https://www.computerhope.com/jargon/d/drawio.htm
[Accessed 4 5 2024].

Simplilearn, 2023. *Data Structure in Python : A Comprehensive Guide.* [Online]
Available at: https://www.simplilearn.com/tutorials/python-tutorial/data-structures
[Accessed 6 5 2024].

10. Appendix

Code for main.py :

main.py

```
# Importing necessary modules
from operation import rent, returnLand
from write import rentingBill, returningBill, changeStatus
from read import displayLandInfo


'''
This serves as the entry point for the Land Rental System. It displays the welcome message,
prompts the user to select an option (Rent, Return, or Exit), and then executes the corresponding action.
'''



print(
    "\n\n------------------------ WELCOME TO TECHNO PROPERTY NEPAL ------------------------\n"
)


displayLandInfo() # Displaying information about lands
```

```
# Prompt user to select an option
print("\nPlease select any option: ")
print("\n1. Rent")
print("\n2. Return")
print("\n3. Exit")

# Continue prompting user until a valid option is selected
while True:

    choice = input("\nWhat's your choice: ").lower()

    if choice == "1" or choice == "rent":
        # Rent land
        data = rent()

        rentingBill(data)

    elif choice == "2" or choice == "return":
        # Return land

        returnLand()



    elif choice == "3" or choice == "exit":
```

```
    # Exit the program
    print("\nThank you for using our Service.")

    exit()

else:
    # Invalid option
    print("\nChoose a valid option.")

    continue



# Ask if the user wants to continue using the system
while True:

    choice = input(
        "\nDo you want to continue running our land rental system ? (y/n): "
    ).lower()

    if choice == "y" or choice == "yes":

        break

    elif choice == "n" or choice == "no":
```

```
            print("\n################################## Thank you for using using our
service. ##################################\n")



            exit()



        else:

            # Invalid answer

            print("\nChoose a valid answer.")



            continue
```

Figure 34:main.py code

Code for operation.py :

operation.py

```
# Importing necessary modules
from read import getLandsData
from write import changeStatus, returningBill
from read import formatData



def rent():
    """
    Function to handle the process of renting land.
    It returns a list containing customer name, phone number, list of rented land IDs, and corresponding durations.
    """

    # Get customer name
    while True:

        name = input("\nEnter customer name: ").title()

        if len(name) < 3:
```

```
        print("\nPlease enter a valid name")

    else:

        break

# Get customer phone number
while True:

    try:

        phone = formatData(input("\nEnter customer phone: "))

        if len(phone) < 10 and int(phone) < 9000000000:

            print("\nEnter a valid phone number")

        else:

            break

    except:

        print("Please enter a valid phone number")
```

```
    # Get lands data
    landsDataDict = getLandsData()


    # renting = True



    # Initializing lists to store rented land IDs and durations
    rentingList = []


    rentingDuration = []


    # Loop to select lands for rent
    while True:


        try:
            # Get land ID
            landID = input("\nEnter kitta no. of the land: ")


            if landID not in landsDataDict:


                print("\nEnter a valid land id.")


                continue


            elif landID in rentingList:
```

```
            print(
                "\nThis land is already added to your invoice. Please select another one."
            )

            continue

        elif formatData(landsDataDict[landID][4]) == "Not Available":

            print("\nThis land is already rented. Please select another one.")

            continue

        else:

            rentingList.append(landID)

    except:

        print("\nEnter a valid land id.")

    # Get renting duration
    while True:

        try:
```

```
            duration = int(input("\nFor how many months he/she wants to rent ? : "))

            if duration < 1 or duration > 360:

                print("\nEnter months between 1 and 360 months.")

                continue

            else:

                rentingDuration.append(duration)

                break

        except:

            print("\nEnter valid number of months.")

    # Asking if more land is to be rented
    choice = input("\nDo he/she wants to rent more land(y/n): ")

    if choice == "yes" or choice == "y":

        continue
```

```
        else:

            break

    # Updating status of rented lands
    for i in range(len(rentingList)):

        changeStatus(rentingList[i], name, rentingDuration[i], "Not Available")

    return [name, phone, rentingList, rentingDuration]




def returnLand():
    """
    Function to handle the process of returning rented land.
    It returns the ID of the land to be returned.
    """

    # Get lands data
    d = getLandsData()

    returningLands = []
```

```
# Loop to select land to return
while True:


    landID = input("\nEnter kitta number of the land to return : ")

    try:

        if formatData(d[landID][4].lower()) == "available":

            print("\nThis land has not been rented yet. Please select another land. ")

            continue

    except:

        print("\nPlease choose a valid kitta number.")

        continue



    if landID not in d:
```

```
        print("\nThis land is not in the list. Please choose a valid kitta number.")

        continue

    sameUser = False

    for each in returningLands:

        name = formatData(d[each][5].lower())
        custName = formatData(d[landID][5].lower())

        if custName != name:

            print("\nThis customer has not rented this land.")

            sameUser = True

    ask = True

    if sameUser == True:

        sameUser = False

        choice = input("\nDo you want to return more land ? (y/n): ")
```

```
        if choice == "yes" or choice == "y":

            continue

        else:

            ask = False

    elif landID in returningLands:

        print("\nThis land is already added to the bill. Please select another land.")

        continue

    else:

        returningLands.append(landID)

    if ask == True:

        choice = input("\nDo you want to return more land(y/n): ").lower()

        if choice == "yes" or choice == "y":

            continue
```

```
            else:

                break

        else:

            break

    returningBill(returningLands)



# returnLand()
```

Figure 35:operation.py code

Code for write.py :

write.py

```
from datetime import datetime, date

from time import localtime, strftime

from random import randint

from read import formatData, getLandsData, showBill


fileName = "land_info.txt"



def rentingBill(data):

    """

    Function to generate and print a renting invoice.

    It will take a list containing customer name, phone number, rented land IDs, and corresponding durations.

    """


    # Extract data from the input list

    name = data[0]

    phone = data[1]

    rentingList = data[2]

    rentingDuration = data[3]


    # Get the current date

    current_date = str(date.today())
```

```
# Get the current time
current_time = strftime("%I:%M:%S", localtime())

# Generate a random invoice ID
invoiceID = str(randint(100000, 900000))

# Construct the invoice name
invoiceName = "Renting-Invoice-" + invoiceID + ".txt"

# Open the invoice file
file = open(invoiceName, "w")

# Write invoice header and customer details
file.write(
    "--------------------------------------- Invoice ---------------------------------------------"
)

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

file.write(
    "    " + formatData("Invoice ID: " + invoiceID, 70) + "Date: " + current_date
)

file.write(
```

```
        "\n\n    " + formatData("Customer Name: " + name, 70) + "Time: " + current_time
    )

    file.write("\n\n    " + formatData("Customer Phone: " + str(phone), 70))

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

    file.write(
        "    SN  Kitta number  Location      Direction  Anna    Rent       Duration     Total
\n\n"
    )

    # Get lands data
    landsDataDict = getLandsData()

    grandTotal = 0

    # Write rented land details to the invoice
    for i in range(len(rentingList)):

        SN = str(i + 1)
        kittaNumber = str(rentingList[i])
```

```
        duration = str(rentingDuration[i])
        location = landsDataDict[str(rentingList[i])][0]
        direction = landsDataDict[str(rentingList[i])][1]
        anna = landsDataDict[str(rentingList[i])][2]
        rent = landsDataDict[str(rentingList[i])][3]

        file.write("    " + formatData(SN, 3))
        file.write(" " + formatData(kittaNumber, 13))
        file.write(" " + formatData(location, 13))
        file.write(" " + formatData(direction, 10))
        file.write(" " + formatData(anna, 7))
        file.write(" " + formatData(rent, 11))
        file.write(" " + formatData(duration, 12))

        total = int(rentingDuration[i]) * int(rent)

        file.write(" " + formatData(str(total), 10))

        file.write("\n")

        grandTotal = grandTotal + total

    # Write grand total to the invoice
    file.write(
        "\n   ------------------------------------------------------------------------------------------   \n"
```

```
    )

    file.write(
        "\n                                                Grand Total: "
        + str(grandTotal)
        + "\n\n"
    )

    # Display the invoice
    file.close()

    showBill(invoiceName)




def returningBill(landsID):
    """
    Function to generate and print a returning invoice.

    it will take the ID of the land being returned.
    """

    d = getLandsData()
```

```
    # Get the current date
    current_date = str(datetime.now().date())

    # Get the current time
    current_time = strftime("%I:%M:%S", localtime())

    # Generate a random invoice ID
    invoiceID = str(randint(100000, 900000))

    # Get customer name from land data
    # print(landsID,landsID[0])
    # print(landsID[0][5])
    name = d[landsID[0]][5]

    # Constructing the invoice name
    invoiceName = "Returning-Bill-" + invoiceID + ".txt"

    # Opening the invoice file
    file = open(invoiceName, "w")

    # Write invoice header and customer details
    file.write(
        "------------------------------------------------- Invoice ----------------------------------------
----------"
    )
```

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


    file.write(
        "    " + formatData("Invoice ID: " + invoiceID, 88) + "Date: " + current_date
    )


    file.write(
        "\n\n    " + formatData("Customer Name: " + name, 88) + "Time: " + current_time
    )


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


    file.write(
        "    Kitta number  Location      Direction  Anna    Rent       Duration     Extra
Duration  Fine      Total       \n\n"
    )


    allTotal = 0


    for each in landsID:
```

```
        kittaNumber = str(each)
        location = str(d[each][0])
        direction = str(d[each][1])
        anna = str(d[each][2])
        rent = int(d[each][3])
        duration = int(d[each][6])

        total = duration * rent

        # calculating extra dureation for fine

        rentedDate = datetime.strptime(d[landsID[0]][7], "%Y-%m-%d")
        current_date = datetime.now()
        date_difference = current_date - rentedDate

        usedMonths = int(date_difference.days / 30)

        extraMonths = usedMonths - duration



        if extraMonths < 0:

            extraMonths = 0
```

```
        fineAmount = (rent * extraMonths) + (10 / 100 * (rent * extraMonths))


        total = total + fineAmount


        file.write("    " + formatData(kittaNumber, 13))
        file.write(" " + formatData(location, 13))
        file.write(" " + formatData(direction, 10))
        file.write(" " + formatData(anna, 7))
        file.write(" " + formatData(str(rent), 11))
        file.write(" " + formatData(str(duration), 12))
        file.write(" " + formatData(str(extraMonths), 15))
        file.write(" " + formatData(str(fineAmount), 9))
        file.write(" " + formatData(str(total), 16) + "\n")


        allTotal = allTotal + total


        changeStatus(each, "None", 0, "Available")



    file.write("\n")


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

```
    file.write(
        "\n                                                    Grand Total: "
        + str(allTotal)
        + "\n\n"
    )

    file.close()

    showBill(invoiceName)


def writeNewData(d):

    file = open(fileName, "w")

    for id, val in d.items():

        file.write("  " + str(id) + "  ")

        for each in val:

            file.write(",  " + str(each) + "  ")

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

```
    file.close()



def changeStatus(id, cust, month, status):
    """
    Function to change the status of a land (Available/Not Available) and update
customer details and renting duration.

    It will take :
        1) id (str): The ID of the land.
        2) cust (str): The name of the customer.
        3) month (int): The renting duration in months.
        4) status (str): The status of the land (Available/Not Available).
    """



    # Get lands data
    d = getLandsData()


    # Update land status, customer details, and renting duration
    d[id][4] = status


    d[id][5] = cust
```

```
    d[id][6] = month


    # Update renting date based on status
    if status == "Available":


        d[id][7] = "Renting Date Here"


    elif status == "Not Available":


        d[id][7] = str(datetime.now().date())


    # Write updated data to the file
    writeNewData(d)
```

Figure 36:write.py code

Code for read.py :

read.py

```
# Defining the file path
filePath = "land_info.txt"



def formatData(string, length=0):
    """
    Function to format string data by removing leading and trailing spaces and adjusting its length.

    It will take :
        string (str): The string data to be formatted.
        length (int, optional): The desired length of the formatted string. Defaults to 0.

    It returns the formatted string.
    """

    # Remove leading and trailing spaces
    if len(string) < 2:

        string = string

    else:
```

```
    while string[0] == " ":

        string = string[1:]

    while string[-1] == " ":

        string = string[:-1]


# Adjust string length
if length == 0:

    return string

elif length > len(string):

    while length != len(string):

        string += " "

    return string

elif length < len(string):
```

```
        return string[0:length]

    return string



def displayLandInfo():
    """
    Function to display information about all lands in a formatted table.
    """

    # Open the file
    file = open(filePath, "r")

    # Read lines from the file
    lines = file.readlines()

    # Print table header
    print(
        "\n---------------------------------- All Lands ----------------------------------"
    )

    print("\n\tKitta  Location          Direction  Area    Rent        Status\n")

    # Iterating through each line
    for i in range(len(lines)):
```

```
        # Removing newline characters and split line into list
        lines[i].replace("\n", "")

        lines[i] = lines[i].split(",")

        # Print formatted data for each land
        print("\t", end="")

        print(formatData(str(lines[i][0]), 6), end="")
        print(" " + formatData(str(lines[i][1]), 18), end="")
        print(" " + formatData(str(lines[i][2]), 10), end="")
        print(" " + formatData(str(lines[i][3]), 7), end="")
        print(" " + formatData(str(lines[i][4]), 12), end="")
        print(" " + formatData(str(lines[i][5]), 15), end="")

        print()
    # Print table footer
    print(
        "\n---------------------------------------------------------------------------"
    )


def getLandsData():
    """
```

```
    Function to read land data from the file and return it as a dictionary.
    It will return dictionary containing land information.
    """

    # Initialize dictionary to store land data
    landsDataDict = {}

    # Open the file
    file = open(filePath, "r")

    # Read lines from the file
    lines = file.readlines()

    # Parse each line and add data to dictionary
    for line in lines:

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

        line = line.split(",")

        key = formatData(line[0])

        val = []

        for i in range(1, len(line)):
```

```
            val.append(formatData(line[i]))

        landsDataDict[key] = val

    return landsDataDict



def showBill(invoiceName):
    """
    Function to display the contents of a billing invoice.

    It will take the name of the invoice file.
    """

    # Open the invoice file
    file = open(invoiceName, "r")

    # Read all lines from the invoice
    allLands = file.readlines()

    print("\n")
    # Print each line of the invoice
    for each in allLands:
```

```
        if "-- Invoice --" in each:

            print("   ", each, end="    ")

        else:

            print(each, end="    ")

    print()
```

Figure 37:read.py code