CS4051NI Fundamentals of Computing

60% Individual Coursework

2023/24 Spring

Student Name: Manish Shrestha

London Met ID: 23050365

College ID: np01cp4a230415

Assignment Due Date: Tuesday, May 7, 2024

Assignment Submission Date: Tuesday, May 7, 2024

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.

Acknowledgement

I feel incredibly fortunate to have completed the coursework, as it required significant guidance and support from various respected individuals for me to succeed. I'm deeply grateful for their direction and assistance, as it's the primary factor behind my achievements. I want to express my appreciation to Mr. Hrishav Tandukar and Mr. Bijay Gautam for providing us with this coursework opportunity.

Table of Contents

Lists of Figures.

List of Tables

1. Introduction

1.1 Python

Python, a dynamic, high-level programming language, is renowned for its object-oriented approach and dynamic semantics. Its extensive array of built-in data structures, along with features like dynamic typing and binding, makes it a favoured choice for swift application development and as a scripting tool to seamlessly integrate existing components. Python prioritizes code readability, thereby reducing the complexity of program maintenance. Its syntax is straightforward and easily comprehensible, fostering quick learning.

Support for packages and modules encourages the creation of modular, reusable code. The Python interpreter and standard library are freely available for all major platforms, enabling unrestricted sharing.

Figure 1: Python

Python's appeal lies in its ability to enhance productivity. Its streamlined edit-test-debug cycle, devoid of a compilation stage, expedites development. Debugging Python programs is straightforward, with errors reported as exceptions rather than obscure segmentation faults. The interpreter provides a detailed stack trace in case of unhandled exceptions. Moreover, Python offers a source-level debugger, empowering developers to

inspect variables, execute expressions, set breakpoints, and step through code systematically. Notably, Python's introspective capabilities are demonstrated by the fact that the debugger itself is implemented in Python. Despite these advanced debugging features, the straightforward addition of print statements remains a commonly used debugging method.

In conclusion, Python's widespread adoption can be attributed to its popularity, versatility, and ease of learning, distinguishing it as a preferred language among programmers compared to alternatives. (Anon., n.d.)

1.2 Goals and Objective

The objectives of this coursework are multifaceted:

- Introducing students to Python's IDLE (Integrated Development and Learning Environment).
- Supporting students in developing a broad spectrum of skills relevant to the subject.
- Deepening comprehension of Python's functions and modules through practical application.
- Facilitating hands-on experience with coding techniques.

2.Discussion and Analysis

2.1 Algorithm

An algorithm is a structured approach to tackling a problem or executing a calculation. It provides a precise set of instructions for carrying out a task step by step, whether in hardware or software processes. Typically, an algorithm starts with initial input and a defined set of instructions for a specific computation, ultimately producing an output upon completion (Gillis, n.d.).

Step 1: START

Step 2: Display Welcome message and address

Step 3: Display the options: Rent Land, Return Land, Exit

Step 4: Input the chosen option

Step 5: Validate the input option

Step 6: If invalid, go back to Step 4; else, proceed based on the chosen option

Step 7: If the chosen option is Rent Land:

Step 7.1: Display available lands

Step 7.2: Input customer details (name and contact)

Step 7.3: Input Land ID

Step 7.4: Check the Land ID

Step 7.5: If ID is valid, input duration of rent

Step 7.6: Validate duration

Step 7.7: Update land status and calculate total cost

Step 7.8: Ask if the customer wants to rent more lands

Step 7.9: If yes, go back to Step 7.1; else, proceed

Step 7.10: Ask if the customer wants shipping

Step 7.11: If yes, create a bill with shipping cost; else, create a bill without shipping cost

Step 7.12: Go back to Step 3

Step 8: If the chosen option is Return Land:

Step 8.1: Display rented lands

Step 8.2: Input customer details (name and contact)

Step 8.3: Input Land ID

Step 8.4: Check the Land ID

Step 8.5: If ID is valid, input actual duration of rent

Step 8.6: Calculate fine (if any) and update land status

Step 8.7: Ask if the customer wants to return more lands

Step 8.8: If yes, go back to Step 8.1; else, proceed

Step 8.9: Print and create a bill with VAT

Step 8.10: Go back to Step 3

Step 9: If the chosen option is Exit:

2.2 Flowchart

A flowchart is a visual representation illustrating the functioning of a system, computer algorithm, or process. It employs various shapes like rectangles, ovals, and diamonds, along with connecting arrows, to denote different types of steps and the flow and sequence between them. Widely utilized across diverse fields, flowcharts serve to analyse, structure, improve, and communicate complex processes through straightforward diagrams (LucidChart, n.d.) .

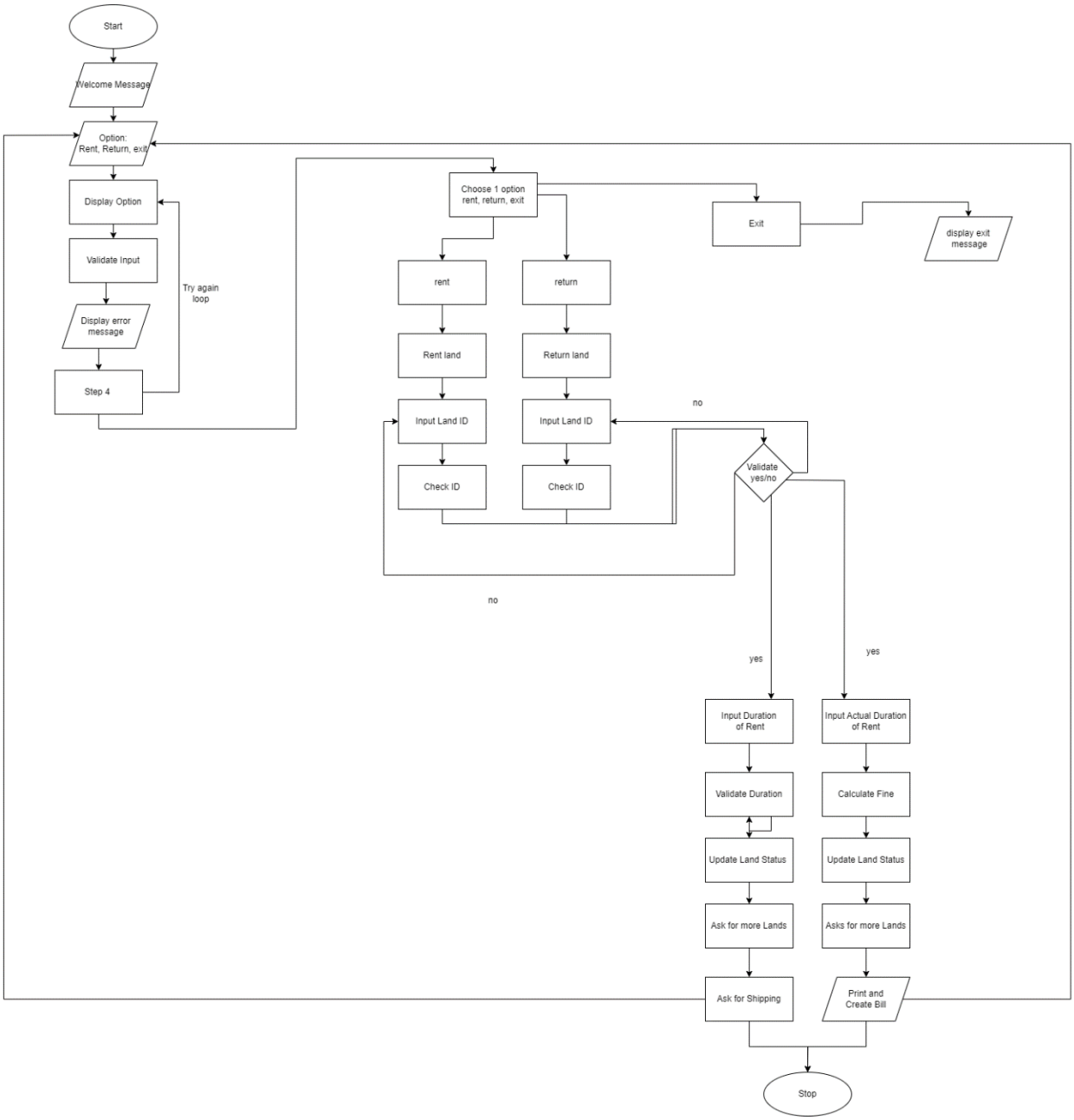


Figure 2: Flow Chart Diagram

3.Pseudocode

Pseudocode offers a way to articulate an algorithm without adhering to strict syntax rules. Acquiring proficiency in reading and writing pseudocode enhances your ability to communicate with fellow programmers, regardless of the differences in programming languages they use (Learn, n.d.)

3.1 Pseudocode for main.py

```
IMPORT datetime module AS date

PRINT "Welcome to the Land Rental System"

DEFINE class Land

    DEFINE __init__ method with parameters plot_id, location, direction, area, price, status

        SET self.plot_id TO plot_id

        SET self.location TO location

        SET self.direction TO direction

        SET self.area TO area

        SET self.price TO price

        SET self.status TO status
```

DEFINE __str__ method

RETURN formatted string of Land details

DEFINE class RentalInvoice

DEFINE __init__ method with parameters customer_name, date_time, rentals, total_amount

SET self.customer_name TO customer_name

SET self.date_time TO date_time

SET self.rentals TO rentals

SET self.total_amount TO total_amount

DEFINE save_invoice method

CREATE filename using customer_name and date_time

OPEN filename for writing AS file

WRITE rental invoice details to file

PRINT message indicating invoice is saved

DEFINE class ReturnInvoice

DEFINE __init__ method with parameters customer_name, date_time, returns, total_fine

SET self.customer_name TO customer_name

SET self.date_time TO date_time

SET self.returns TO returns

SET self.total_fine TO total_fine

DEFINE save_invoice method

```
        CREATE filename using customer_name and date_time
        OPEN filename for writing AS file
        WRITE return invoice details to file
        PRINT message indicating invoice is saved
DEFINE class LandManagementSystem
    DEFINE __init__ method
        CALL read_land_file method and assign result to self.lands
    DEFINE read_land_file method
        OPEN land_details file for reading AS file
        READ lines from file and split into list of lists
        CONVERT each sublist into Land objects and return as list
    DEFINE write_land_file method
        OPEN lands.txt file for writing AS file
        LOOP through lands list
            WRITE each land's details to file
    DEFINE print_lands_data method with parameter available (default True)
        PRINT "Available Lands:" if available ELSE "Rented Lands:"
        LOOP through lands list
            IF land is available AND available is True OR land is not available AND available
is False
                PRINT land details
```

```
DEFINE rent_status_lands method
    SET rent_status TO True
    INITIALIZE total_amount TO 0
    INITIALIZE rentals TO empty list
    PROMPT user for customer_name
    WHILE rent_status is True
        CALL print_lands_data method
        PROMPT user for plot_id
        TRY
            PROMPT user for duration as integer
            SET found TO False
            LOOP through lands list
                IF land plot_id matches input plot_id AND land is available
                    SET land status to Not Available
                    CALCULATE cost
                    ADD cost to total_amount
                    APPEND rental details to rentals list
                    SET found to True
                    BREAK LOOP
            IF found is False
                PRINT error message
```

```
        EXCEPT ValueError
            PRINT error message for invalid input
        PROMPT user to rent more lands
        IF user input is not 'yes'
            SET rent_status to False
    IF rentals list is not empty
        SET date_now TO current date and time in specified format
        CREATE RentalInvoice object with customer_name, date_now, rentals, total_amount
        CALL save_invoice method on RentalInvoice object
        CALL write_land_file method
        PRINT success message
    ELSE
        PRINT "No lands were rented."
DEFINE return_land method
    SET returning TO True
    INITIALIZE total_fine TO 0
    INITIALIZE returns TO empty list
    PROMPT user for customer_name
    WHILE returning is True
        CALL print_lands_data method with available=False
```

```
PROMPT user for plot_id to return
TRY
    PROMPT user for actual_duration as integer
    PROMPT user for contract_duration as integer
    SET found TO False
    LOOP through lands list
        IF land plot_id matches input plot_id AND land is not available
            SET land status to Available
            CALCULATE cost
            CALCULATE fine
            ADD fine to total_fine
            APPEND return details to returns list
            SET found to True
            BREAK LOOP
    IF found is False
        PRINT error message
EXCEPT ValueError
    PRINT error message for invalid input
PROMPT user to return more lands
IF user input is not 'yes'
    SET returning to False
```

```
    IF returns list is not empty
        SET date_now TO current date and time in specified format
        CREATE ReturnInvoice object with customer_name, date_now, returns, total_fine
        CALL save_invoice method on ReturnInvoice object
        CALL write_land_file method
        PRINT success message
    ELSE
        PRINT "No lands were returned."
DEFINE display_all_lands method
    CALL print_lands_data method with available=True
    CALL print_lands_data method with available=False
DEFINE main method
    WHILE True
        PRINT menu options
        PROMPT user to choose an option
        IF user input is '1'
            CALL rent_status_lands method
        ELSE IF user input is '2'
            CALL return_land method
        ELSE IF user input is '3'
            CALL display_all_lands method
```

```
        ELSE IF user input is '4'
            PRINT farewell message
            BREAK LOOP
        ELSE
            PRINT error message for invalid option
IF script is executed directly
    INITIALIZE LandManagementSystem object AS land_system
    CALL main method
```

3.2 Pseudocode for operation.py

```
DEFINE function read_plot_file
        SET plot_details as "./plots/plots.txt"
        OPEN plot_details in read mode as file
        READ lines from file and STORE as plots
        CLOSE file
        SET plots as a list comprehension where each element is a split of a line in plots by comma and space
        RETURN a list comprehension where each element is an instance of Plot class created with the unpacked elements of plots


DEFINE function write_plot_file
        PARAMETERS: plots
```

```
    OPEN "./plots/plots.txt" in write mode as file

    FOR each plot in plots DO

        WRITE a line composed of the joined elements of plot separated by comma
and space

    CLOSE file
```

3.3 Pseudocode for read.py

```
DEFINE function Plot

    PARAMETERS: plot_id, location, direction, area, price, status

        SET self.plot_id to plot_id

        SET self.location to location

        SET self.direction to direction

        SET self.area to area

        SET self.price to price

        SET self.status to status


DEFINE function str

    RETURN string concatenation of "Plot ID: ", self.plot_id, ", Location: ", self.location,
    ", Direction: ", self.direction, ", Area (annas): ", self.area, ", Price (per month): Rs
    ", self.price
```

3.4 Pseudocode of write.py

```
    DEFINE function save_rent_invoice

        PARAMETERS: customer_name, date_time, rentals, total_amount
```

```
        SET filename as f"./rent_invoice/{customer_name.replace(' ', '')}{date_time.replace(':', '-')}.txt"
        OPEN filename in write mode as file
        WRITE "Techno Property Nepal Multiple Plot Rental Invoice\n"
        WRITE "-----------------------------------------------\n"
        WRITE f"Date/Time: {date_time}\n"
        WRITE f"Customer Name: {customer_name}\n"
        FOR each rental in rentals DO
            WRITE f"Plot ID: {rental['plot_id']}, Location: {rental['location']}, "
            WRITE f"Direction: {rental['direction']}, Area (annas): {rental['area']}, "
            WRITE f"Duration (months): {rental['duration']}, Cost: Rs {rental['cost']}\n"
            WRITE f"Total Amount: Rs {total_amount}\n"
            PRINT f"Invoice saved as {filename}"


DEFINE function save_return_invoice
    PARAMETERS: customer_name, date_time, returns, total_fine
        SET filename as f"./return_invoice/{customer_name.replace(' ', '')}{date_time.replace(':', '-')}.txt"
        OPEN filename in write mode as file
        WRITE "Techno Property Nepal Multiple Plot Return Invoice\n"
```

```
WRITE "-----------------------------------------------\n"
WRITE f"Date/Time: {date_time}\n"
WRITE f"Customer Name: {customer_name}\n"
FOR each return_detail in returns DO
    WRITE f"Plot ID: {return_detail['plot_id']}, Location: {return_detail['location']}, "
    WRITE f"Direction: {return_detail['direction']}, Area (annas): {return_detail['area']}, "
    WRITE f"Contract Duration (months): {return_detail['contract_duration']}, "
    WRITE f"Actual Duration (months): {return_detail['actual_duration']}, "
    WRITE f"Cost: Rs {return_detail['cost']}, Fine: Rs {return_detail['fine']}\n"
    WRITE f"Total Fine: Rs {total_fine}\n"
    PRINT f"Invoice saved as {filename}"
```

4.Data Structure

A data structure, in computer memory organization, is a framework implemented within a programming language to organize and manage data efficiently. Python offers two primary types of built-in data structures: mutable and immutable.

Mutable data structures allow for the addition, removal, or modification of elements. Python's mutable data structures include lists, dictionaries, and sets.

In contrast, immutable data structures cannot be altered once created. In Python, tuples represent the sole fundamentally built-in immutable data structure (Sannikov, 2022).

4.1 Dictionary

Dictionaries are those mutable data structures that include a collection of keys and values inside them. It is used to quickly access specific data associated to a unique key. We can easily create a dictionary by curly brackets “{ }” or by the constructor dict() (Sannikov, 2022).

4.2 Lists

Python lists are dynamic and mutable arrays that hold an ordered sequence of items. Lists can accommodate various data types and objects, such as integers, strings, and even functions, all within the same structure. The indexing of list elements starts from 0, allowing access to different entries using integer indices. Operations like addition, removal, and modification of elements are supported in lists. Lists can be created either by using square brackets, "[ ]", or the list() constructor (Sannikov, 2022).

```
def rent_status_lands(self):
    rent_status = True
    total_amount = 0
    rentals = []
    customer_name = input("Enter your name: ")
```

Figure 3: using Lists

4.3 Sets

Sets in Python are dynamic and mutable collections that contain unique, immutable elements. Their defining characteristic is that they do not allow duplicates, making them effective for removing duplicates from lists. Sets excel at quickly determining whether a specific element is present within them. Sets can be created using curly brackets, "{}", or

by employing the set() constructor. Unlike dictionaries, sets do not contain key-value pairs (Sannikov, 2022).

For instance, consider the set of fruits: fruits = {"Apple", "Banana", "Mango", "Grape", "Papaya"}.

Now the Immutable data structures are as follows:

4.4 Tuples

Similarly, to lists, tuples in Python can store ordered collections of elements. The key distinction lies in their immutability; once created, tuples cannot be altered. Tuples are suitable when a data structure needs to remain unchanged after creation. Additionally, tuples can serve as dictionary keys, provided all their elements are immutable. Tuples can be created using round brackets, "()", or by utilizing the tuple() constructor (Sannikov, 2022).

For example, consider the tuple of animals: animals = ("Monkey", "Cow", "Lion", "Deer").

4.5 String

A string in Python is a data structure used to represent a sequence of characters. Strings are commonly employed for storing and manipulating textual data, as well as representing various types of information such as names and addresses. In Python, there isn't a distinct data type for a single character; instead, a single character is represented as a string with a length of 1. To access a specific character within a string, square brackets are used to reference its position within the string (geeksforgeeks, 2023)

For instance, consider the usage of strings in Python: print("Hello World"), str().

4.6 Integers

In Python, an integer stands as a fundamental data structure, representing whole numbers that can be either positive or negative without decimal points. When a number includes decimal places, it transforms into a floating-point, also known as a float (Fadeyi, 2022).

```
while rent_status:
    self.print_lands_data()
    plot_id = input("\nEnter the land ID: ")
    try:
        duration = int(input("\nEnter the duration of the rent in months: "))
        found = False
        for land in self.lands:
            if land.plot_id == plot_id and land.status == "Available":
                land.status = "Not Available"
                cost = int(land.price) * duration
                total_amount += cost
                rentals.append({
                    'plot_id': land.plot_id,
                    'location': land.location,
                    'direction': land.direction,
                    'area': land.area,
                    'duration': duration,
                    'cost': cost
```

Figure 4: Using Integer

4.7 Boolean

Booleans are fundamental data structures in Python that yield a true or false value when an expression, variable, or value is assessed. The bool() function evaluates the parameter or variable as either true or false (Fadeyi, 2022).

```
def write_land_file(self):
    with open("./lands/lands.txt", "w") as file:
        for land in self.lands:
            file.write(', '.join([land.plot_id, land.location, land.direction, land.area, land.price, land.status]) +

def print_lands_data(self, available=True):
    print("\nAvailable Lands:" if available else "\nRented Lands:")
    for land in self.lands:
        if (land.status == "Available" and available) or (land.status == "Not Available" and not available):
            print(land)
def rent_status_lands(self):
    rent_status = True
    total_amount = 0
    rentals = []
    customer_name = input("Enter your name: ")

    while rent_status:
        self.print_lands_data()
        plot_id = input("\nEnter the land ID: ")
        try:
            duration = int(input("\nEnter the duration of the rent in months: "))
            found = False
```

Figure 5: Using Boolean

4.8 Float

In Python, a value type is called a float. When used on an integer or string argument, float() returns a floating point number or a decimal point. For example, 1.23 is a float (Ojaokomo, 2022)

5. Program

A set of instruction that is done by computer to complete any task is called program. Here, I have designed a Land Rental System using Python IDLE.

5.1 Implementation of the Land Rental System

The Land Rental System is a Python program designed to manage the renting and returning of lands in a property rental agency.

Classes:

Land: Represents individual land properties with attributes like ID, location, direction, area, price, and status (available or not).

RentalInvoice: Handles the creation and saving of rental invoices, including customer name, date/time of rental, rented lands, and total amount.

ReturnInvoice: Manages the creation and saving of return invoices, containing details such as customer name, date/time of return, returned lands, and total fines incurred.

LandManagementSystem: Serves as the core system for managing lands, including reading and writing land data, renting lands, returning lands, and displaying available/rented lands.

Main Features:

Rent Land: Allows customers to rent available lands by specifying the land ID and rental duration. Automatically generates rental invoices and updates the status of rented lands.

Return Land: Enables customers to return previously rented lands by providing the land ID, actual rental duration, and contract duration. Generates return invoices and updates the availability of returned lands.

Display Available and Rented Lands: Shows the current status of all lands, separating available and rented lands.

This system streamlines the management of land rentals, providing a user-friendly interface for customers to rent and return lands while maintaining accurate records of transactions.

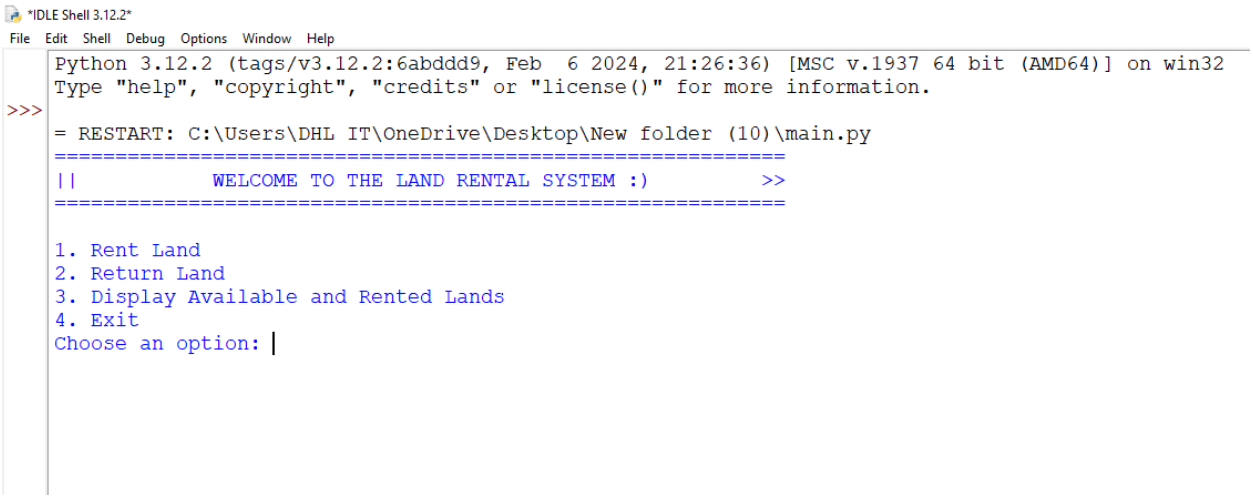


Figure 6: Implementation of program

5.2 The Rent and Return of the Land

The process of renting and returning land involves several steps within the Land Rental System. To rent land, the system prompts the user to input their name and then displays a list of available lands. The user selects the land they wish to rent by providing its ID and specifying the rental duration in months. Upon confirmation, the system updates the status of the chosen land to "Not Available," calculates the total cost based on the rental duration and land price, and generates an invoice. This invoice includes details such as the customer's name, rental date, rented land information, and total amount due.

Conversely, returning land involves the user providing their name and specifying the land they wish to return by its ID. They must also input the actual duration of the rent and the duration as per the contract. The system then updates the status of the returned land to "Available," calculates any fines incurred based on the deviation from the contract duration, generates a return invoice, and updates the system's records accordingly. Both processes ensure accurate tracking of land rentals and returns while providing customers with detailed invoices for their transactions.

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 1
Enter your name: manish

Available Lands:
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 106, Location: Butwal, Direction: South, Area (annas): 6, Price (per month): Rs 60000
Land ID: 107, Location: Dharan, Direction: West, Area (annas): 9, Price (per month): Rs 90000

Enter the land ID: 102

Enter the duration of the rent in months: 5

Would you like to rent more lands? (yes/no): n
```

Figure 7: Renting a land

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 1
Enter your name: manish

Available Lands:
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 106, Location: Butwal, Direction: South, Area (annas): 6, Price (per month): Rs 60000
Land ID: 107, Location: Dharan, Direction: West, Area (annas): 9, Price (per month): Rs 90000

Enter the land ID: 102

Enter the duration of the rent in months: 5

Would you like to rent more lands? (yes/no): n
Invoice saved as ./rent_invoice/manish_2024-05-06 20-49-43.txt

                ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                Thank you for choosing our lands.
                        All selected lands have been rented successfully.
                    Please check the rent_invoice folder for your invoice.
                +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 8: Showing renting Details

```
manish_2024-05-06 04-40-35 - Notepad
File  Edit  Format  View  Help
TechnoPropertyNepal Multiple Land Rental Invoice
----------------------------------------------
Date/Time: 2024-05-06 04:40:35
Customer Name: manish
Kitta Number: 102, Location: Bhaktapur, Direction: East, Area (annas): 5, Duration (months): 5, Cost: Rs 22500000
Total Amount: Rs 22500000
```

Figure 9: Displaying bill in text file.

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 2
Enter your name: manish

Rented Lands:
Land ID: 101, Location: Kathmandu, Direction: North, Area (annas): 4, Price (per month): Rs 50000
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 103, Location: Lalitpur, Direction: South, Area (annas): 10, Price (per month): Rs 100000
Land ID: 104, Location: Bhaktapur, Direction: West, Area (annas): 8, Price (per month): Rs 80000
Land ID: 105, Location: Biratnagar, Direction: East, Area (annas): 7, Price (per month): Rs 70000
Enter the land ID of the land to return:
```

Figure 10: Returning the land

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 2
Enter your name: manish

Rented Lands:
Land ID: 101, Location: Kathmandu, Direction: North, Area (annas): 4, Price (per month): Rs 50000
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 103, Location: Lalitpur, Direction: South, Area (annas): 10, Price (per month): Rs 100000
Land ID: 104, Location: Bhaktapur, Direction: West, Area (annas): 8, Price (per month): Rs 80000
Land ID: 105, Location: Biratnagar, Direction: East, Area (annas): 7, Price (per month): Rs 70000
Enter the land ID of the land to return: 102
Enter the actual duration of the rent in months: 2
Enter the duration as per contract in months: 5
Do you want to return more lands? (yes/no): n
Invoice saved as ./return_invoice/manish_2024-05-06_21_06_09.txt
All selected lands have been returned successfully.

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: |
```

Figure 11: Displaying the returning details.

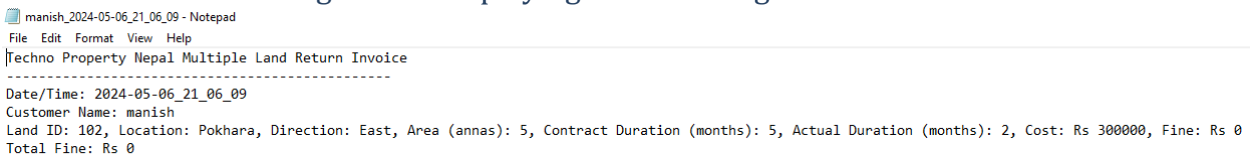


Figure 12: Displaying the land returning invoice in text file.

5.3 Creation of Text file

After execution of program the bill are printed in .txt format and are stored in same folder which consists program as shown below. Here, two bills are printed which are of renting a land and returning a land respectively.

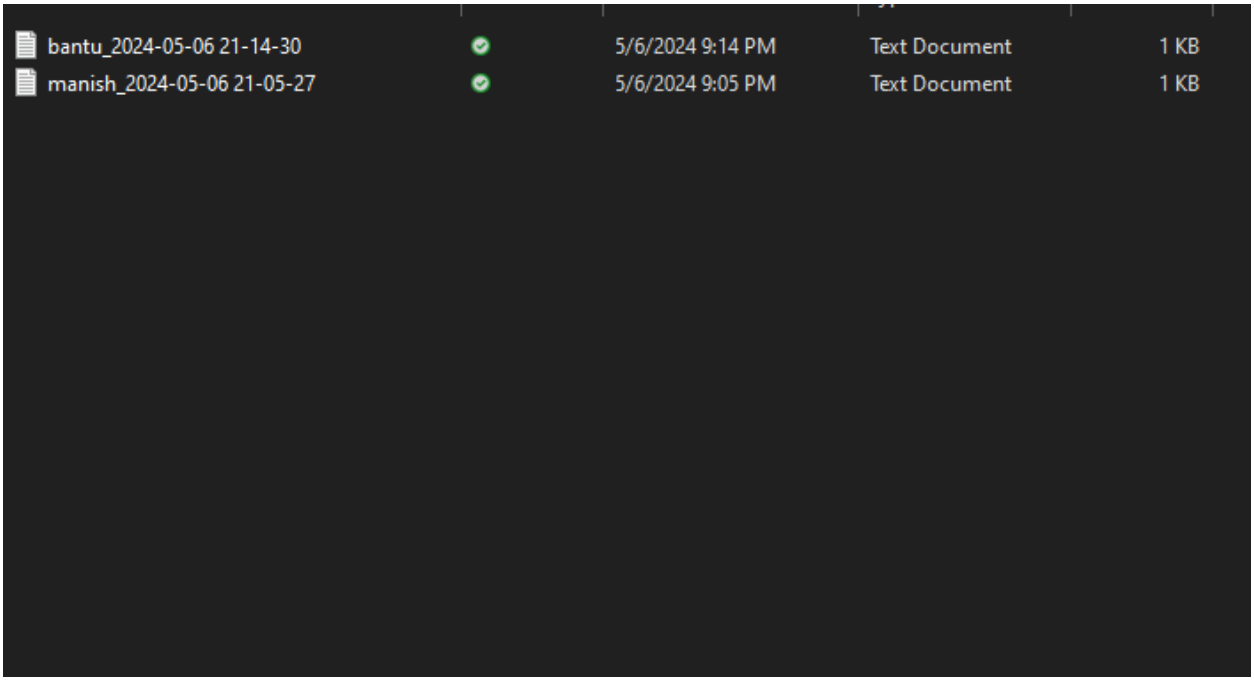
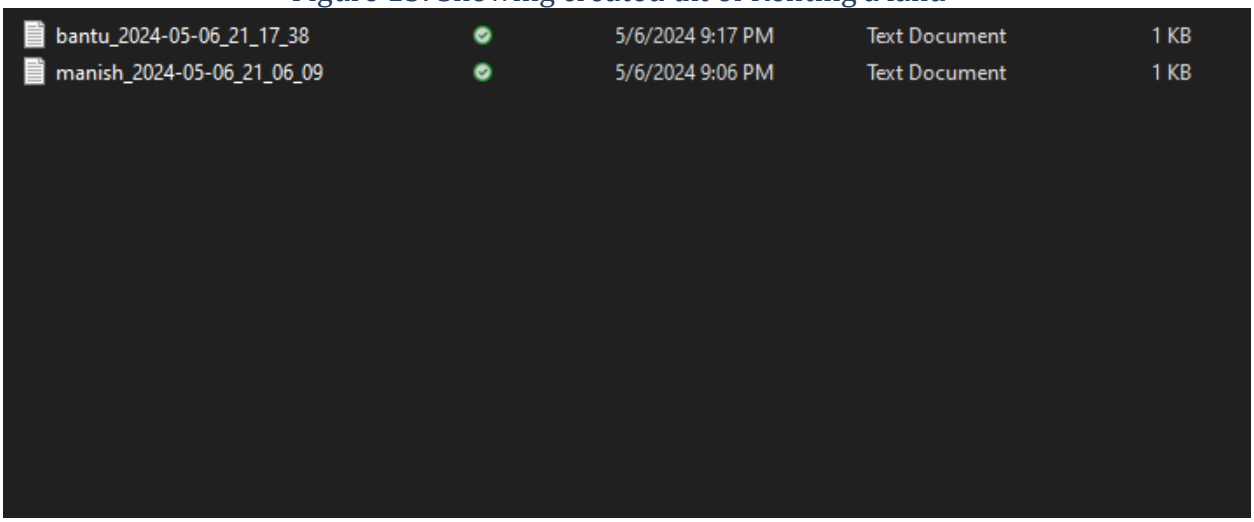


Figure 13: Showing created txt of Renting a land

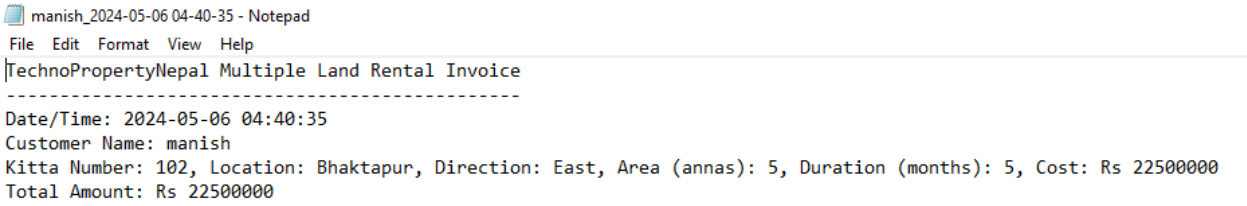


bantu_2024-05-06_21_17_38	5/6/2024 9:17 PM	Text Document	1 KB
manish_2024-05-06_21_06_09	5/6/2024 9:06 PM	Text Document	1 KB

Figure 14: Showing created txt of Returning a land

5.4 Opening the text file and showing the bill

After the creation of text file, the bills are created as shown in below pictures.



manish_2024-05-06 04-40-35 - Notepad

File Edit Format View Help

```
TechnoPropertyNepal Multiple Land Rental Invoice
-----------------------------------------------
Date/Time: 2024-05-06 04:40:35
Customer Name: manish
Kitta Number: 102, Location: Bhaktapur, Direction: East, Area (annas): 5, Duration (months): 5, Cost: Rs 22500000
Total Amount: Rs 22500000
```

Figure 15: Showing the Bills of renting the land.

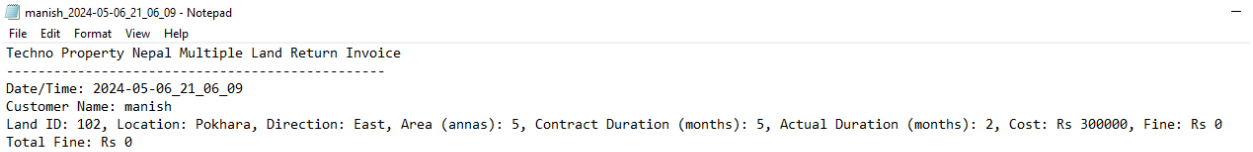


Figure 16: Showing the Bills of returning the land.

5.5 Termination of the program

This python program can easily be terminated through user input 4. The user can

terminate the program by choosing the fourth option when shown.

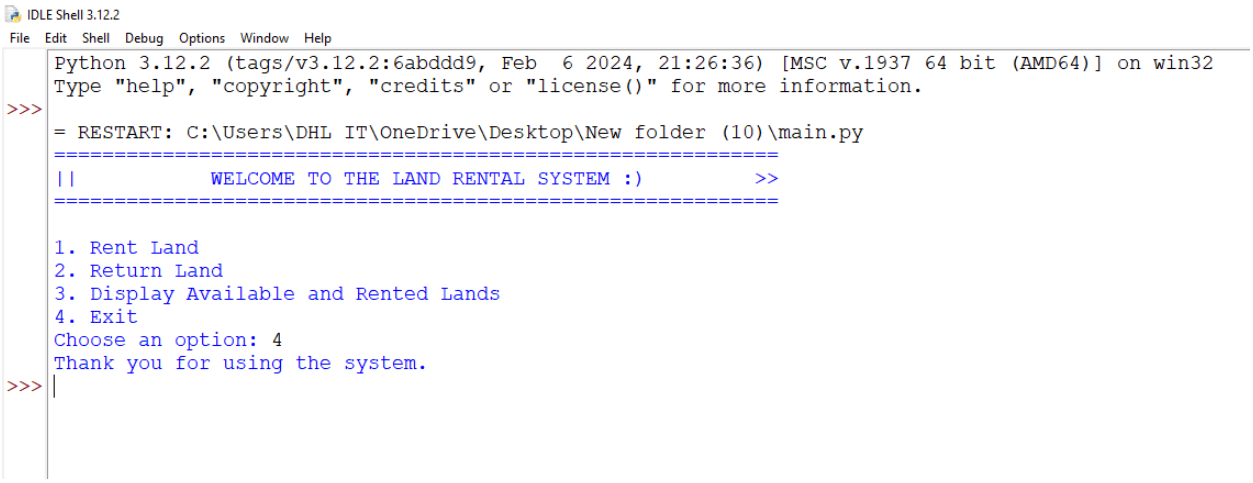


Figure 17: Termination of the Program.

6. Testing

Test 1: Show the implementation of try, except and display message

Objective	To show the implementation of try, except

Action	Enter invalid input to the given options
Expected Result	An appropriate message will be shown and the program will be restarted.
Actual Result	An appropriate message will be shown the program was restarted.
Conclusion	The test was successful.

Table 1: To show the implementation of try, except

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 5
Invalid option, please try again.

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: |
```

Figure 18: To show the implementation of try, except

Test 2: Selection rent and return of lands.

a. Provide the negative value as input.

Objective	To provide the negative value as input
Action	Enter negative value while asking the land ID.
Expected Result	The selected land is unavailable or the land ID is invalid will be displayed.

Actual Result	The selected land is unavailable or the land ID is invalid was displayed.
Conclusion	The test was successful.

Table 2: To provide the negative value as input

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 1
Enter your name: manish

Available Lands:
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 106, Location: Butwal, Direction: South, Area (annas): 6, Price (per month): Rs 60000
Land ID: 107, Location: Dharan, Direction: West, Area (annas): 9, Price (per month): Rs 90000

Enter the land ID: -101

Enter the duration of the rent in months: 5

The selected land is unavailable or the land ID is invalid.

Would you like to rent more lands? (yes/no): |
```

Figure 19: Providing the negative value as input

Provide the non-existed value as input

Objective	To provide none existed value as input
Action	Enter non-existed value as input
Expected Result	The selected land is unavailable or the land ID is invalid will be displayed.
Actual Result	The selected land is unavailable or the land ID is invalid was displayed.
Conclusion	The test was successful.

Table3: To provide none existed value as input

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 1
Enter your name: manish

Available Lands:
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 106, Location: Butwal, Direction: South, Area (annas): 6, Price (per month): Rs 60000
Land ID: 107, Location: Dharan, Direction: West, Area (annas): 9, Price (per month): Rs 90000

Enter the land ID: 109

Enter the duration of the rent in months: 5

The selected land is unavailable or the land ID is invalid.

Would you like to rent more lands? (yes/no): |
```

Figure 20: Providing non-existed value as input

b. Provide the negative value as input.

Objective	To provide the negative value as input
Action	Enter negative value while asking the land ID.
Expected Result	Invalid land ID or the land is not currently rented will be displayed.
Actual Result	Invalid land ID or the land is not currently rented was displayed.
Conclusion	The test was successful.

Table 4: Providing negative value as input.

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 2
Enter your name: manish

Rented Lands:
Land ID: 101, Location: Kathmandu, Direction: North, Area (annas): 4, Price (per month): Rs 50000
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 103, Location: Lalitpur, Direction: South, Area (annas): 10, Price (per month): Rs 100000
Land ID: 104, Location: Bhaktapur, Direction: West, Area (annas): 8, Price (per month): Rs 80000
Land ID: 105, Location: Biratnagar, Direction: East, Area (annas): 7, Price (per month): Rs 70000
Enter the land ID of the land to return: -101
Enter the actual duration of the rent in months: 5
Enter the duration as per contract in months: 5
Invalid land ID or the land is not currently rented.
Do you want to return more lands? (yes/no):
```

Figure 21:providing negative value as input

Provide the non-existed value as input

Objective	To provide none existed value as input
Action	Enter non-existed value as input
Expected Result	Invalid land ID or the land is not currently rented will be displayed.
Actual Result	Invalid land ID or the land is not currently rented was displayed.
Conclusion	The test was successful.

Table 5: Providing non existed value as input.

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 2
Enter your name: manish

Rented Lands:
Land ID: 101, Location: Kathmandu, Direction: North, Area (annas): 4, Price (per month): Rs 50000
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 103, Location: Lalitpur, Direction: South, Area (annas): 10, Price (per month): Rs 100000
Land ID: 104, Location: Bhaktapur, Direction: West, Area (annas): 8, Price (per month): Rs 80000
Land ID: 105, Location: Biratnagar, Direction: East, Area (annas): 7, Price (per month): Rs 70000
Enter the land ID of the land to return: 109
Enter the actual duration of the rent in months: 5
Enter the duration as per contract in months: 5
Invalid land ID or the land is not currently rented.
Do you want to return more lands? (yes/no):
```

Figure 22: providing non existed value as input.

Test 3: File generation of renting of land.

Objective	To generate renting of lands.
Action	All the required input is given in order to rent the lands.
Expected Result	The lands will be added and output will be displayed and will create a text file of displayed bill.
Actual Result	The lands will be added and output was displayed and a text file of displayed bill was created.
Conclusion	The test was successful.

Table 6: To generate renting of lands.

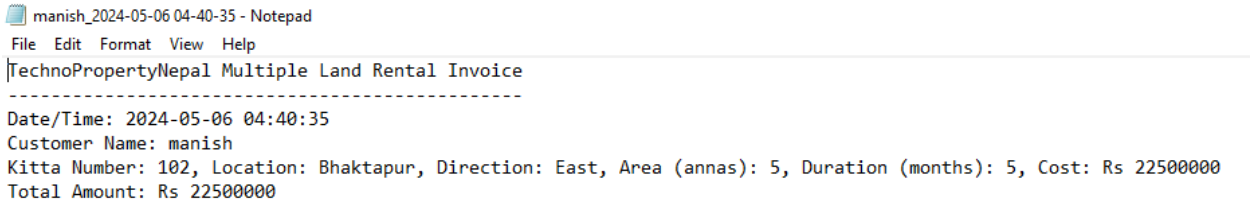
```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 1
Enter your name: manish

Available Lands:
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 106, Location: Butwal, Direction: South, Area (annas): 6, Price (per month): Rs 60000
Land ID: 107, Location: Dharan, Direction: West, Area (annas): 9, Price (per month): Rs 90000

Enter the land ID: 102

Enter the duration of the rent in months: 5

Would you like to rent more lands? (yes/no): n
```

Figure 23: Displaying details.

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 1
Enter your name: manish

Available Lands:
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 106, Location: Butwal, Direction: South, Area (annas): 6, Price (per month): Rs 60000
Land ID: 107, Location: Dharan, Direction: West, Area (annas): 9, Price (per month): Rs 90000

Enter the land ID: 102

Enter the duration of the rent in months: 5

Would you like to rent more lands? (yes/no): n
Invoice saved as ./rent_invoice/manish_2024-05-06 20-49-43.txt

                ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                              Thank you for choosing our lands.
                       All selected lands have been rented successfully.
                    Please check the rent_invoice folder for your invoice.
                +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 24: Renting lands.

2


manish_2024-05-06 04-40-35 - Notepad

File Edit Format View Help

```
TechnoPropertyNepal Multiple Land Rental Invoice
-----------------------------------------------
Date/Time: 2024-05-06 04:40:35
Customer Name: manish
Kitta Number: 102, Location: Bhaktapur, Direction: East, Area (annas): 5, Duration (months): 5, Cost: Rs 22500000
Total Amount: Rs 22500000
```

Figure 25: Displaying rented bill created in text file

Test 4: File generation of returning of land.

Objective	To generate returning of lands.
Action	All the required input is given in order to return the lands.
Expected Result	The lands will be added and output will be displayed and will create a text file of returned lands.
Actual Results	The lands will be added and output was displayed and a text file of returned was created.
Conclusion	The test was successful.

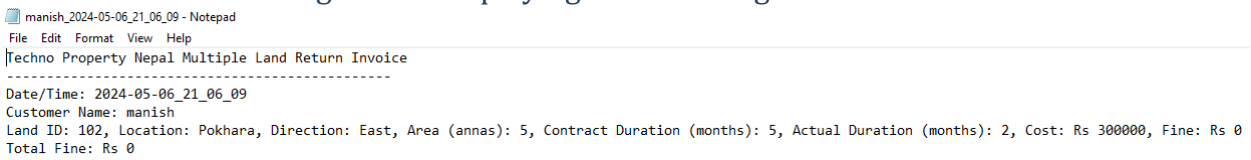
Table 7: To generate returning of lands.

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 2
Enter your name: manish

Rented Lands:
Land ID: 101, Location: Kathmandu, Direction: North, Area (annas): 4, Price (per month): Rs 50000
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 103, Location: Lalitpur, Direction: South, Area (annas): 10, Price (per month): Rs 100000
Land ID: 104, Location: Bhaktapur, Direction: West, Area (annas): 8, Price (per month): Rs 80000
Land ID: 105, Location: Biratnagar, Direction: East, Area (annas): 7, Price (per month): Rs 70000
Enter the land ID of the land to return:
```

Figure 26: Displaying details

```
*IDLE Shell 3.12.2*
File  Edit  Shell  Debug  Options  Window  Help
>>>
= RESTART: C:\Users\DHL IT\OneDrive\Desktop\New folder (10)\main.py
============================================================
||           WELCOME TO THE LAND RENTAL SYSTEM :)         >>
============================================================

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: 2
Enter your name: manish

Rented Lands:
Land ID: 101, Location: Kathmandu, Direction: North, Area (annas): 4, Price (per month): Rs 50000
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Price (per month): Rs 60000
Land ID: 103, Location: Lalitpur, Direction: South, Area (annas): 10, Price (per month): Rs 100000
Land ID: 104, Location: Bhaktapur, Direction: West, Area (annas): 8, Price (per month): Rs 80000
Land ID: 105, Location: Biratnagar, Direction: East, Area (annas): 7, Price (per month): Rs 70000
Enter the land ID of the land to return: 102
Enter the actual duration of the rent in months: 2
Enter the duration as per contract in months: 5
Do you want to return more lands? (yes/no): n
Invoice saved as ./return_invoice/manish_2024-05-06_21_06_09.txt
All selected lands have been returned successfully.

1. Rent Land
2. Return Land
3. Display Available and Rented Lands
4. Exit
Choose an option: |
```

Figure 27: Displaying the returning lands.


manish_2024-05-06_21_06_09 - Notepad

File Edit Format View Help

```
Techno Property Nepal Multiple Land Return Invoice
-----------------------------------------------
Date/Time: 2024-05-06_21_06_09
Customer Name: manish
Land ID: 102, Location: Pokhara, Direction: East, Area (annas): 5, Contract Duration (months): 5, Actual Duration (months): 2, Cost: Rs 300000, Fine: Rs 0
Total Fine: Rs 0
```

Figure 28: Displaying returned bill created in text file

7.Conclusion

This coursework project involved the development of a Land Rental System for Techno Property Nepal. The system allows users to rent lands, generate rental invoices, return lands, and generate return invoices. Python was the chosen programming language for the implementation of the system.

The primary objectives of the project included:

- Managing land information including ID, location, direction, area, price, and availability status.
- Handling rental transactions by allowing users to rent lands for a specified duration.
- Generating rental invoices for customers based on their rental transactions.
- Handling returns transactions by allowing users to return lands and calculating any fines incurred.
- Generating return invoices for customers based on their return transactions.
- Ensuring data integrity and consistency throughout the rental and return processes.

The development process encountered several challenges, such as reading and processing land information from text files and implementing robust transaction functionalities. These challenges were overcome through meticulous research, problem-

solving, and leveraging Python's file handling capabilities and string manipulation techniques.

Key features of the system include:

- Comprehensive documentation outlining functionality, design decisions, and encountered issues.
- Modular design facilitating the management of complexity and improving maintainability.
- Thorough testing, complemented by relevant screen captures, ensuring correctness and functionality.
- Integration of input validation mechanisms to ensure the system only accepts valid and reliable data, minimizing transaction errors.
- Accurate updating of the land status and generation of detailed invoices for rental and return transactions.

In summary, this coursework project provided valuable practical experience in applying fundamental computing concepts to develop a functional application. It enhanced programming skills in Python, sharpened problem-solving abilities, and fostered a deeper understanding of fundamental computing principles and their real-world applications.

8. Bibliography

Anon., n.d. *Lucidchart.* [Online]
Available at: https://www.lucidchart.com/pages/what-is-a-flowchart-tutorial
Anon., n.d. *Python.* [Online]

Available at: https://www.python.org/doc/essays/blurb/

Fadeyi, A., 2022. *top coder.* [Online]

Available at: https://www.topcoder.com/thrive/articles/data-structures-and-algorithms-in-python?utm_source=thrive&utm_campaign=thrive-feed&utm_medium=rss-feed

geeksforgeeks, 2023. *geeksforgeeks.* [Online]

Available at: https://www.geeksforgeeks.org/python-string/

Gillis, A. S., n.d. *Algorithm.* [Online]

Available at: https://www.techtarget.com/whatis/definition/algorithm

Learn, F., n.d. *FutureLearn.* [Online]

Available at: https://www.futurelearn.com/info/courses/block-to-text-based-programming/0/steps/39492

LucidChart, n.d. *LucidChart.* [Online]

Available at: https://www.lucidchart.com/pages/what-is-a-flowchart-tutorial

Ojaokomo, 2022. *HubsSpot.* [Online]

Available at: https://blog.hubspot.com/website/what-is-python#:~:text=A%20float%20is%20a%20type,for%20any%20floating%2Dpoint%20nu

Sannikov, A., 2022. *Data Quest.* [Online]

Available at: https://www.dataquest.io/blog/data-structures-in-python/

9. Appendix

9.1 Appendix of main.py

```
from datetime import datetime as date
```

```
print("==============================================================")
print("||          WELCOME TO THE LAND RENTAL SYSTEM :)        >>")
print("==============================================================")


class Land:
    def __init__(self, plot_id, location, direction, area, price, status):
        self.plot_id = plot_id
        self.location = location
        self.direction = direction
        self.area = area
        self.price = price
        self.status = status

    def __str__(self):
        return f"Land ID: {self.plot_id}, Location: {self.location}, Direction: {self.direction},
Area (annas): {self.area}, Price (per month): Rs {self.price}"


class RentalInvoice:
    def __init__(self, customer_name, date_time, rentals, total_amount):
        self.customer_name = customer_name
        self.date_time = date_time
```

```
        self.rentals = rentals
        self.total_amount = total_amount


    def save_invoice(self):
        filename = f"./rent_invoice/{self.customer_name.replace(' ', '_')}_{self.date_time.replace(':', '-')}.txt"
        with open(filename, "w") as file:
            file.write("Techno Property Nepal Multiple Land Rental Invoice\n")
            file.write("--------------------------------------------------\n")
            file.write(f"Date/Time: {self.date_time}\n")
            file.write(f"Customer Name: {self.customer_name}\n")
            for rental in self.rentals:
                file.write(f"Land ID: {rental['plot_id']}, Location: {rental['location']}, ")
                file.write(f"Direction: {rental['direction']}, Area (annas): {rental['area']}, ")
                file.write(f"Duration (months): {rental['duration']}, Cost: Rs {rental['cost']}\n")
            file.write(f"Total Amount: Rs {self.total_amount}\n")
        print(f"Invoice saved as {filename}")


class ReturnInvoice:
    def __init__(self, customer_name, date_time, returns, total_fine):
        self.customer_name = customer_name
```

```
        self.date_time = date_time
        self.returns = returns
        self.total_fine = total_fine

    def save_invoice(self):
        filename = f"./return_invoice/{self.customer_name.replace(' ', '_')}_{self.date_time.replace(':', '-')}.txt"
        with open(filename, "w") as file:
            file.write("Techno Property Nepal Multiple Land Return Invoice\n")
            file.write("-----------------------------------------------\n")
            file.write(f"Date/Time: {self.date_time}\n")
            file.write(f"Customer Name: {self.customer_name}\n")
            for return_detail in self.returns:
                file.write(f"Land ID: {return_detail['plot_id']}, Location: {return_detail['location']}, ")
                file.write(f"Direction: {return_detail['direction']}, Area (annas): {return_detail['area']}, ")
                file.write(f"Contract Duration (months): {return_detail['contract_duration']}, ")
                file.write(f"Actual Duration (months): {return_detail['actual_duration']}, ")
                file.write(f"Cost: Rs {return_detail['cost']}, Fine: Rs {return_detail['fine']}\n")
            file.write(f"Total Fine: Rs {self.total_fine}\n")
        print(f"Invoice saved as {filename}")
```

```
class LandManagementSystem:
    def __init__(self):
        self.lands = self.read_land_file()

    def read_land_file(self):
        land_details = "./lands/lands.txt"
        with open(land_details, "r") as file:
            lands = [line.strip().split(', ') for line in file.readlines()]
        return [Land(*land) for land in lands]

    def write_land_file(self):
        with open("./lands/lands.txt", "w") as file:
            for land in self.lands:
                file.write(', '.join([land.plot_id, land.location, land.direction, land.area, land.price, land.status]) + '\n')

    def print_lands_data(self, available=True):
        print("\nAvailable Lands:" if available else "\nRented Lands:")
        for land in self.lands:
            if (land.status == "Available" and available) or (land.status == "Not Available" and not available):
```

```
            print(land)
def rent_status_lands(self):
    rent_status = True
    total_amount = 0
    rentals = []
    customer_name = input("Enter your name: ")


    while rent_status:
        self.print_lands_data()
        plot_id = input("\nEnter the land ID: ")
        try:
            duration = int(input("\nEnter the duration of the rent in months: "))
            found = False
            for land in self.lands:
                if land.plot_id == plot_id and land.status == "Available":
                    land.status = "Not Available"
                    cost = int(land.price) * duration
                    total_amount += cost
                    rentals.append({
                        'plot_id': land.plot_id,
                        'location': land.location,
```

```
                'direction': land.direction,
                'area': land.area,
                'duration': duration,
                'cost': cost
            })
            found = True
            break
        if not found:
            print("\nThe selected land is unavailable or the land ID is invalid.")
    except ValueError:
        print("\nInvalid input. Please enter a valid number for the duration.")
    more = input("\nWould you like to rent more lands? (yes/no): ").lower()
    if more != 'yes':
        rent_status = False

if rentals:
    date_now = date.now().strftime("%Y-%m-%d %H:%M:%S")
    rental_invoice = RentalInvoice(customer_name, date_now, rentals, total_amount)
    rental_invoice.save_invoice()
    self.write_land_file()
    print("""
```

```
            ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        Thank you for choosing our lands.
                    All selected lands have been rented successfully.
                  Please check the rent_invoice folder for your invoice.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
            """)
        else:
            print("No lands were rented.")


    def return_land(self):
        returning = True
        total_fine = 0
        returns = [ ]
        customer_name = input("Enter your name: ")


        while returning:
            self.print_lands_data(available=False)
            plot_id = input("Enter the land ID of the land to return: ")
            try:
                actual_duration = int(input("Enter the actual duration of the rent in months: "))
```

```
        contract_duration = int(input("Enter the duration as per contract in months: "))
        found = False
        for land in self.lands:
            if land.plot_id == plot_id and land.status == "Not Available":
                land.status = "Available"
                cost = int(land.price) * contract_duration
                fine = (actual_duration - contract_duration) * int(land.price) * 1.5 if actual_duration > contract_duration else 0
                total_fine += fine
                returns.append({
                    'plot_id': land.plot_id,
                    'location': land.location,
                    'direction': land.direction,
                    'area': land.area,
                    'actual_duration': actual_duration,
                    'contract_duration': contract_duration,
                    'cost': cost,
                    'fine': fine
                })
                found = True
                break
```

```
            if not found:
                print("Invalid land ID or the land is not currently rented.")
        except ValueError:
            print("Invalid input. Please enter a valid number for durations.")

        more = input("Do you want to return more lands? (yes/no): ").lower()
        if more != 'yes':
            returning = False

    if returns:
        date_now = date.now().strftime("%Y-%m-%d_%H_%M_%S")
        return_invoice = ReturnInvoice(customer_name, date_now, returns, total_fine)
        return_invoice.save_invoice()
        self.write_land_file()
        print("All selected lands have been returned successfully.")
    else:
        print("No lands were returned.")

def display_all_lands(self):
    self.print_lands_data(available=True)
    self.print_lands_data(available=False)
```

```
    def main(self):
        while True:
            print("\n1. Rent Land\n2. Return Land\n3. Display Available and Rented Lands\n4. Exit")
            choice = input("Choose an option: ")
            if choice == '1':
                self.rent_status_lands()
            elif choice == '2':
                self.return_land()
            elif choice == '3':
                self.display_all_lands()
            elif choice == '4':
                print("Thank you for using the system.")
                break
            else:
                print("Invalid option, please try again.")

if __name__ == "__main__":
    land_system = LandManagementSystem()
    land_system.main()
```

9.2 Appendix of operation.py

```
from datetime import datetime as date


class FileOperations:
    @staticmethod
    def read_plot_file():
        plot_details = "./plots/plots.txt"
        with open(plot_details, "r") as file:
            plots = [line.strip().split(', ') for line in file.readlines()]
        return [Plot(*plot) for plot in plots]


    @staticmethod
    def write_plot_file(plots):
        with open("./plots/plots.txt", "w") as file:
            for plot in plots:
                file.write(', '.join([plot.land_id, plot.location, plot.direction, plot.area, plot.price,
plot.status]) + '\n')
```

9.3 Appendix of read.py

```
class Plot:

    def __init__(self, plot_id, location, direction, area, price, status):

        self.plot_id = plot_id

        self.location = location

        self.direction = direction

        self.area = area

        self.price = price

        self.status = status



    def __str__(self):

        return f"Plot ID: {self.plot_id}, Location: {self.location}, Direction: {self.direction},
Area (annas): {self.area}, Price (per month): Rs {self.price}"
```

9.4 Appendix of write.py

```
class RentalInvoice:

    def __init__(self, customer_name, date_time, rentals, total_amount):

        self.customer_name = customer_name

        self.date_time = date_time

        self.rentals = rentals

        self.total_amount = total_amount
```

```
    def save_invoice(self):
        filename = f"./rent_invoice/{self.customer_name.replace(' ', '_')}_{self.date_time.replace(':', '-')}.txt"
        with open(filename, "w") as file:
            file.write("Techno Property Nepal Multiple Plot Rental Invoice\n")
            file.write("-----------------------------------------------\n")
            file.write(f"Date/Time: {self.date_time}\n")
            file.write(f"Customer Name: {self.customer_name}\n")
            for rental in self.rentals:
                file.write(f"Plot ID: {rental['plot_id']}, Location: {rental['location']}, ")
                file.write(f"Direction: {rental['direction']}, Area (annas): {rental['area']}, ")
                file.write(f"Duration (months): {rental['duration']}, Cost: Rs {rental['cost']}\n")
            file.write(f"Total Amount: Rs {self.total_amount}\n")
        print(f"Invoice saved as {filename}")


class ReturnInvoice:
    def __init__(self, customer_name, date_time, returns, total_fine):
        self.customer_name = customer_name
        self.date_time = date_time
        self.returns = returns
        self.total_fine = total_fine
```

```
    def save_invoice(self):
        filename = f"./return_invoice/{self.customer_name.replace(' ', '_')}_{self.date_time.replace(':', '-')}.txt"
        with open(filename, "w") as file:
            file.write("Techno Property Nepal Multiple Plot Return Invoice\n")
            file.write("-----------------------------------------------\n")
            file.write(f"Date/Time: {self.date_time}\n")
            file.write(f"Customer Name: {self.customer_name}\n")
            for return_detail in self.returns:
                file.write(f"Plot ID: {return_detail['plot_id']}, Location: {return_detail['location']}, ")
                file.write(f"Direction: {return_detail['direction']}, Area (annas): {return_detail['area']}, ")
                file.write(f"Contract Duration (months): {return_detail['contract_duration']}, ")
                file.write(f"Actual Duration (months): {return_detail['actual_duration']}, ")
                file.write(f"Cost: Rs {return_detail['cost']}, Fine: Rs {return_detail['fine']}\n")
            file.write(f"Total Fine: Rs {self.total_fine}\n")
        print(f"Invoice saved as {filename}")
```