CS4051NI Fundamentals of Computing

60% Individual Coursework

2023/24 Spring

Student Name: Tenji Sherpa

London Met ID: 23048523

College ID: Np01nt4a230174

Assignment Due Date:

Assignment Submission Date: Click or tap to enter a date.

Word Count: 3020

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Introduction about the coursework

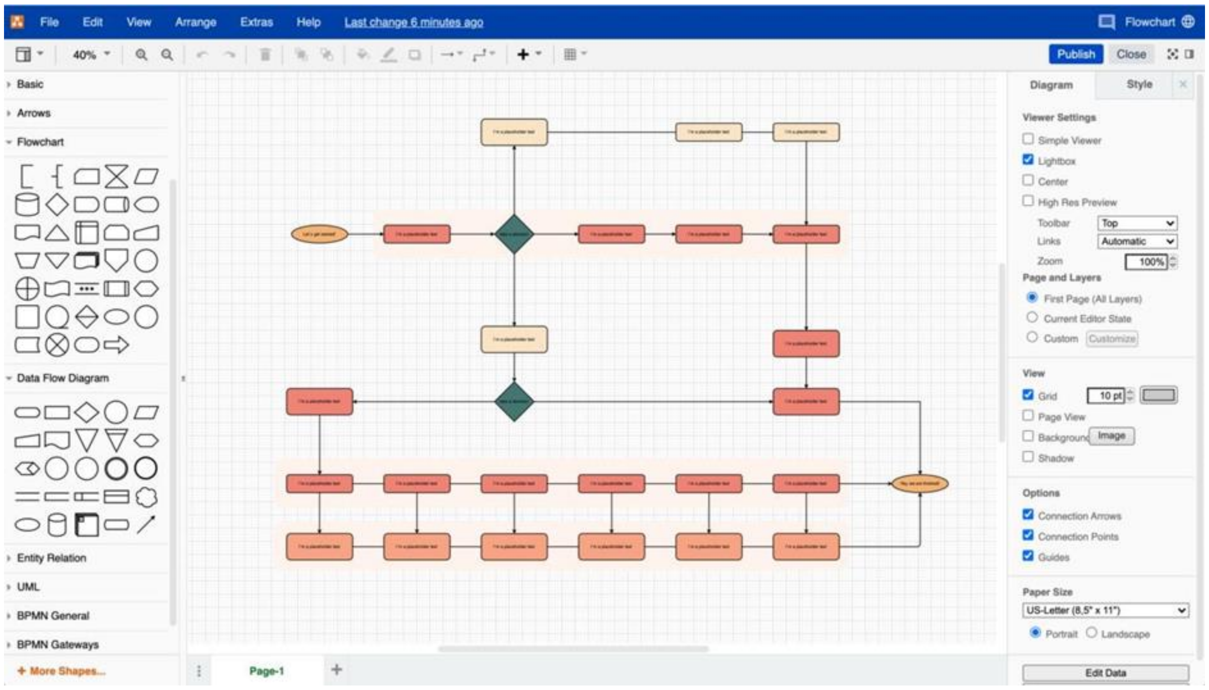
This coursework is an individual project assigned by Islington College, which is affiliated with London Metropolitan University of London. This coursework is all aboutmaking an application for maintaining information for Develop a Land Rental System TechnoPropertyNepal, a private company with an array of land holdings spread over multiple places, aims to optimise its operations by putting in place a strong Land Rental System. The goal of this system is to make it easier to rent and monitor accessible land parcels while maintaining responsibility and openness throughout the entire process.

The Land Rental System is intended to meet the unique requirements of TechnoPropertyNepal, which provides clients with land parcels for a range of uses on a contractual basis. This system is essential for recording rental durations, executing rental agreements, managing land availability, creating billing, and keeping an accurate record of all transactions. TechnoPropertyNepal uses technology to increase productivity, reduce human error, and offer an easy-to-use experience for the business and its customers.

1.2 Tools used for the development of the Project

1.2.1 Draw.io

Draw.io is the popular diagramming tool used for creating various types of diagrams, including flowcharts, class diagrams, ER-Diagram and many more. It is totally free and can be used to make our projects and coursework more attractive. Draw.io is widely used by individuals, teams, and organizations for creating various diagrams for various purposes such as software design, project management etc. (Draw.io, 2023)

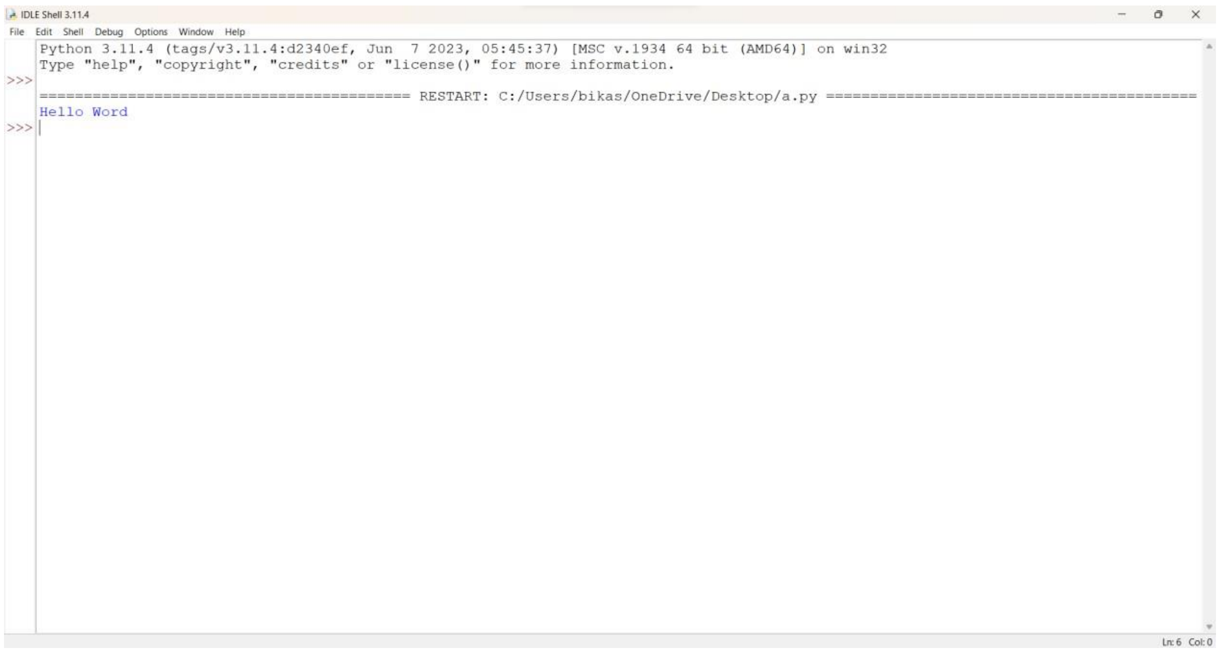


Viewer Settings
Simple Viewer
Lightbox ✓
Center
High Res Preview
Toolbar: Top
Links: Automatic
Zoom: 100%
Page and Layers
● First Page (All Layers)
Current Editor State
Custom Customize
View
Grid ✓ 10 pt
Page View
Background Image
Shadow
Options
Connection Arrows ✓
Connection Points ✓
Guides ✓
Paper Size: US-Letter (8,5" x 11")
● Portrait Landscape

Figure 1 draw.i

1.2.2 Idle

IDLE stands for Integrated Development and Learning Environment which is usedfor the python language. It includes two window types, the Shell window, and the Editor window. IDLE comes with an interactive Python shell that allows the user to execute program line by line. In a code editor, it includes features like syntax highlighting, automatic indentation, auto completion, and other features. It also includes customization features such as font size, color themes, indentation preferences to match your code.



```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======================================== RESTART: C:/Users/bikas/OneDrive/Desktop/a.py ========================================
Hello Word
>>>
```

Figure 2.Idle

1.2.3 Microsoft Word

Microsoft Word is a word processing program which was released in 1983 and is currently used by millions of people every day across the world. It is used to create simple and complex documents. This program can run on multiple platforms like Windows, macOS etc. We can do various things in word like add headers and footers, change font styles, page formatting and many more. We can also open andedit external PDF files as our choice. We can also insert images, video etc. MS Word can be used to create letters, business cards, bills etc. MS Word has a builtindictionary for spell checking; misspelled words are marked with a red squiggly underline. MS Word offers text-level features such as bold, underline, italic and strike-through, and page-level features such as indentation, paragraphing, an justification. Word is compatible with many other programs, the most common beingthe other members of the Office suite. (Microsoft Word, 2023)

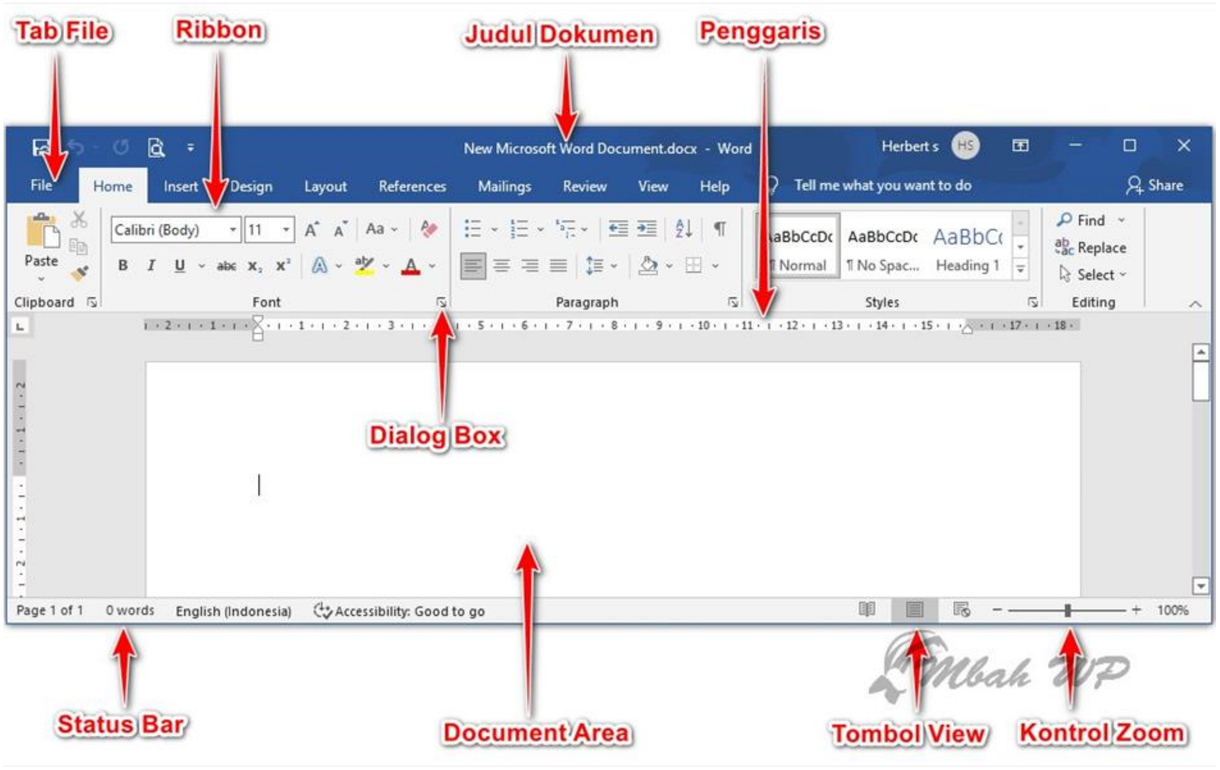


Figure 3 Microsoft Word (Microsoft Word, 2022)

2. Algorithm

Python algorithms are a sequence of steps that are carried out to find the answer to a certain problem. Algorithms can be implemented in a variety of programming languages because they are not language-specific. Algorithms are not written according to any set of rules. They depend on resources and have different issues, yet they use some common code elements like loops and flow-control (if-else) (do, while, for). We will briefly go through Tree Traversal, Sorting, Searching, and Graph Algorithms in the sections that follow (upGrad, 2020)

Algorithm of Land Renting System:
Step 1 : Start

Step 2: Import the datetime.

Step 3: Define Function to Read Land details.

Step 4: Opening the file 'land_details-txt' in read mode using a context manager.

Step 5: Iterate through each line in the file.

Step 6: End of Function Read Land Details.

Step 7: Define Function to display land details in a table.

Step 8: Opening the file 'land_details-txt' in read mode using a context manager.

Step 9: Print the table header.

Step 10: Split each line by the comma and space (', ') delimiter.

Step 11: The file (header) is split into column names.

Step 12: Loop iterates over each column name in the header list.

Step 13: If the last character of the column name is a newline character (\n).

Step 14: If the last character is a newline, this removes it. The slicing operation moves the last character from the string.

Step 15: prints the cell content followed by a tab character \t instead of a newline character. Using end =\t ensures that the cells are printed in a tab-separated format on the same line.

Step 16: A newline is printed to move to the next line after printing the header

Step 17: Call the rent Land details from a file.

Step 18: Enter an infinite loop to display a menu and handle user choices until the user chooses to exit.

Step19: Update the availability status.

Step20: Now we updated lines into list.

Step21: Then, rented the land details into will store the details of the first land found with the status rented.

Step 22: The line starts a loop that iterates over each string in the lines list.

Step 23: Each line is split into parts using a comma as the delimiter, resulting in a list called parts. Each element in parts corresponds to a specific piece of information about the land.

Step24: The line checks if the first element in parts matches the given rent id.

Step25: This land with the given rent id is found, land found is set to True.

Step26: The last element in parts tothe availability status is Available.

Step 27: Initialize availability status is updated to Not Available.

Step 28: The parts list is joined back into a single string using commas, and the result is stored in rent land details.

Step 29: The function returns the updated_lines, rented_land_details and a Boolean land_found.

Step30: Define Main Function named main.

Step 31: Call the function read_land_details() to read land details from a file.

Step 32: Enter an infinite loop to display a menu and handle user choices until the user chooses to exit.

Step33: If the user chooses option 1, filter and display available lands from rent_land.

Step34: If the user chooses option 2, filter and display not available lands from rent_land.

Step35: Iterate through each land dictionary in rent_land. ,For each rent_land, convert its attributes to strings and concatenate them with commas and spaces to form a single line.

Step 36: Import the datetime module.

Step 37: Calculate the total amount by summing the product of each land's price and the duration.

Step 38: Initialize rent_note with a string "Rent Invoice\n\n".

Step 39: Iterate through each land in rented_lands,For each land, append its details (kitta number, city, direction, area) to rent_note using string formatting.

Step 40: Append customer name, current date and time, duration, and total amount to rent_note using string formatting.

Step 41: Generate a unique file name for the rent invoice using the current date and time.

Step42: Open a new file with the generated file name in write mode,write the rent note to the file and after that close the file.

Step 43: Print a message indicating successful generation of the rent invoice along with the file name.

Step44: End of Function.

Step45: Call the function read_land_details() to read land details from a file.

Step 46: Enter an infinite loop to display a menu and handle user choices until the user chooses to exit.

Step 47: If the user chooses option 1, filter and display lands from read_and.

Step 48: If the user chooses option 2, prompt for customer name, duration, and kitta numbers of lands to rent then check if the selected lands are available, generate rent invoices, update land status, and update land details in the file.

Step 49: If the user chooses option 3, prompt for customer name, duration, and kitta numbers of lands to return then check if the selected lands are available, generate return invoices, update land status, and update land details in the file.

Step 50: If the user chooses option 4, print a message and exit the system.

Step 51: If the user enters an invalid choice, prompt them to enter a valid option.

Step 52: If the script is run as the main program, call the main function.

3. Pseudocode

The pseudocode written in Python resembles an algorithmic representation of the underlying code. This indicates that a code cannot be immediately drafted when it is expected to be formulated. Prior to being formalized into a real code, the code must first be created into a Python pseudocode. A Python pseudocode is created using this technique of creating an algorithmic representation that resembles an English sentence. The Python pseudocode is a representation of code that lacks syntax, to put it simply. Thus, there is no code in the Python pseudocode. The computational logic must be accurately reflected in the Python pseudocode. Each line of the Python pseudocode must be able to be translated into real code correspondingly. Python pseudocode makes it simple for non-technical people to understand the actual code. These are the main benefits of programming with Python pseudocode. The main definition to remember in plain English is that Python pseudocode is a representation of code without syntax (EDUCBA, 2022).

Pseudocode of Land Rent System

Main Program (Main.py)

```
DECLARE main()
    DISPLAY welcome message and menu options

   Create Function (rent_land_details)

    WHILE TRUE
        PROMPT user for choice

        IF choice is "1" THEN
            show_land_details(lines)
        ELSE IF choice is "2" THEN
            lines = rent_land(lines)
        ELSE IF choice is "3" THEN
```

```
            lines = return_land(lines)
        ELSE IF choice is "4" THEN
            EXIT program
        ELSE
            DISPLAY "Invalid input, please try again."
        END IF
    END WHILE
END FUNCTION
```

Read Operations

```
FUNCTION read_land_details()
    Open 'land_details.txt' in read mode
    Read all lines from file
    Close file
    RETURN lines
END FUNCTION

FUNCTION show_land_details(lines)
    Open 'Land_details.txt' in read mode
    Read all lines from file
    Close file

    PRINT table header (first line)

    FOR EACH line in lines (excluding header)
        PRINT formatted line data
    END FOR
END FUNCTION
```

Write Operations

```
FUNCTION generate_rent_invoice(rented_lands, customer_name, duration)
```

```
    DECLARE total_amount
    Create rent_note string with invoice details
DECLARE unique filename based on current timestamp
    DECLARE rent_note to file
    DISPLAY success message
END FUNCTION

FUNCTION generate_return_invoice(customer_name, phone, address, time1, time2,
returned_lands, fine, total_price)
    Create return_note string with invoice details
DECLARE unique filename based on current timestamp
    DECLARE return_note to file
    DISPLAY success message
END FUNCTION
```

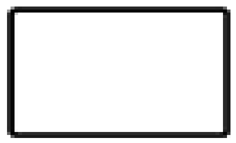
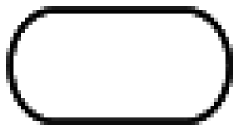
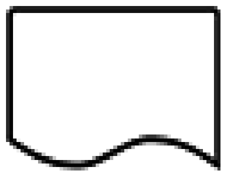
Land Operations

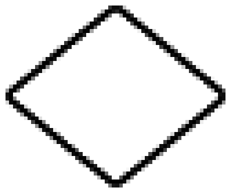
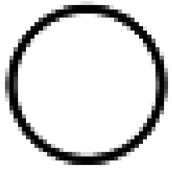
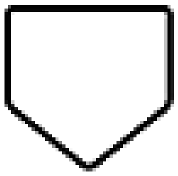
```
FUNCTION rent_land(lines)
    DECLARE user for land ID to rent

    FOR EACH line in lines
        IF land ID matches AND land is available THEN
            Update land status to "Not Available"
            Collect renter's information
            generate_rent_invoice()
            RETURN updated lines
        END IF
    END FOR

    DISPLAY error message if land not found or unavailable
    RETURN original lines
END FUNCTION
```

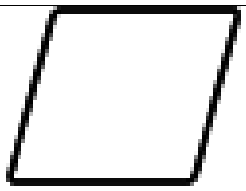
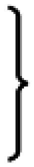
```
FUNCTION return_land(lines)
    DECLARE user for land ID to return

    FOR EACH line in lines
        IF land ID matches AND land is not available THEN
            Update land status to "Available"
            Collect returner's information
            DECLARE fine (if applicable)
            generate_return_invoice()
            RETURN updated lines
        END IF
    END FOR

    DISPLAY error message if land not found or already available
    RETURN original lines
END FUNCTION
```

4. Flowchart

4.1 Commonly used symbols in detailed flowcharts

Flowchart Symbol	Name	Description
	Process symbol	This form, also known as a "Activity Symbol," symbolizes a procedure, an action, or a function. It is the flowcharting symbol that is most frequently used.
	Start/End symbol	This symbol, often known as the "Terminator Symbol," denotes the beginning, middle, and end of a path. frequently has "Start" or "End" inside the form.
	Document symbol	represents a specific document's input or output. Receiving a report, an email, or an order are a few examples of input. An example of an output that uses a document symbol is the creation of a memo, letter, or presentation.

	Decision symbol	A question that needs to be answered, usually with a yes/no or true/false response. Depending on the subsequent outcome or implications, the flowchart path may subsequently diverge into distinct branches.
	Connector symbol	This symbol links various items on a single page and is typically used in more intricate charts.
	Off-Page Connector/Link symbol	This symbol, which joins disparate elements on many pages in complex charts, is frequently accompanied by the page number for convenient reference.
	Input/Output symbol	This shape, which is also known as the "Data Symbol," depicts resources that have been

		used or generated as well as data that is available for input or output. Although the paper tape symbol can also be used to represent input and output, it is no longer widely used when drawing flowcharts.
	Comment/Note symbol	This symbol adds necessary clarification or remarks within the designated range when used in conjunction with context. It could also be linked to the pertinent area of the flowchart by a dashed line (lucidchart, 2022).

5. Data Structures

List, set, tuples, and dictionary are some of the fundamental data structures used in Python. Every data structure is distinctive in its own way. Data structures serve as "containers" for organizing and categorizing data (CFI, 2021).

Data structures serve as a means of organizing and storing data. It is a method of setting up data on a computer to make it easily accessible and up to date. There are two types of data structures:

- Primitive data type
- Non-primitive data type

a. Primitive data type

Fundamental or primitive data types are the most fundamental sorts of data that are used to represent many types of data in programming. These data types include just the most basic and pure data values.

Integers, floats, booleans, and strings are the four basic data types in the Python programming language (CodeSansar, 2022).

- Integer data type
  In Python, an integer is a whole number with zero, a positive or a negative sign, and limitless precision, such as 0, 100, or -10. In Python, the following literals for integers are acceptable.
- String data type
  String is an immutable sequence data type in Python. It is the collection of Unicode characters enclosed in one, two, or three quotations.
- Float data type
  Any real number that has a floating-point representation and a fractional component that is represented by a decimal symbol or scientific notation, such as 1.23 or 3.4556789e2.

- **Boolean data type**

Data that has the predetermined values True or False. The letters "T" and "F" are capitalized. Python will throw an exception for the Boolean values true and false since they are invalid.

b. Non-Primitive data types

The best data format for your project should be chosen based on your requirements and project. For instance, you can use the Array data structure if you want to keep data in memory in a sequential manner (programiz, 2022).

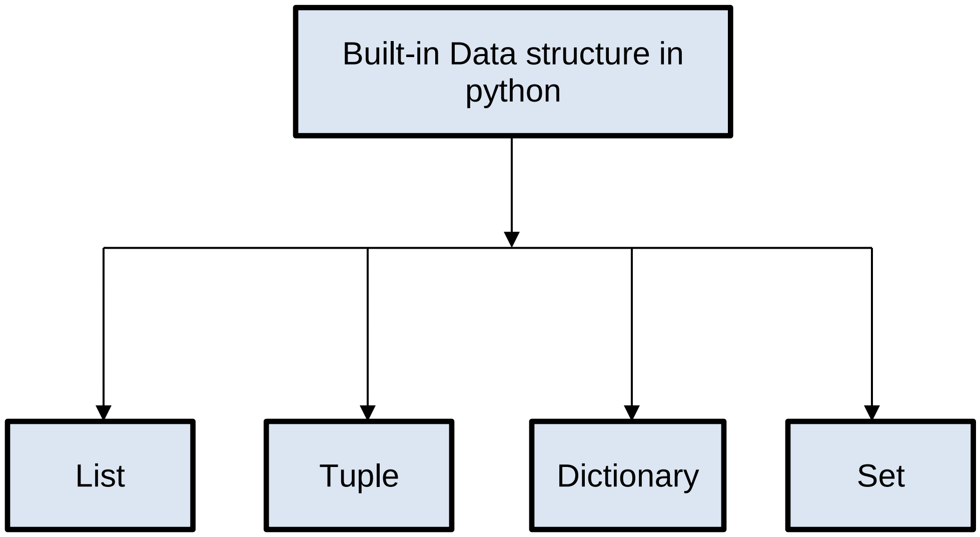


Figure 4:Built-in Data Structure in Python

The mutability and order of the data structures are different. The term "mutability" describes an object's capacity for modification after creation. Immutable objects cannot be changed after being formed, whereas mutable objects can be changed, added to, or removed after being created. In this sense, order has to do with whether an element may be accessed from its place (CFI, 2021).

Example: - tuple1 = (1,2,4,5)

List

Python uses lists to store the sequential order of different kinds of data. Because Python lists are changeable types, we can change their elements after they've been generated. Although

Python has six data types that may hold sequences, the list is the most popular and dependable form.

An assortment of values or things of various categories can be referred to as a list. The list's items are surrounded in square brackets [] and separated by commas (,) (javatpoint, 2022).

Tuples:

Lists and tuples are similar, but tuples are immutable. Tuples are also declared in parenthesis rather than square brackets, unlike Lists. An element cannot be deleted, reassigned, or altered after it has been defined in a Tuple, according to the property of immutability. It makes sure that the data structure's declared values are not changed or overridden (upGrad, 2020).

Dictionaries:

Key-value pairs make up dictionaries. An item is identified by its "key," while its "value" is used to store its worth. The key and its value are separated by a colon. Commas are used to divide the pieces, while curly brackets are used to encapsulate the whole thing. Although the values can be of any type, the keys must be immutable (either integers, Strings, or tuples) (upGrad, 2020).

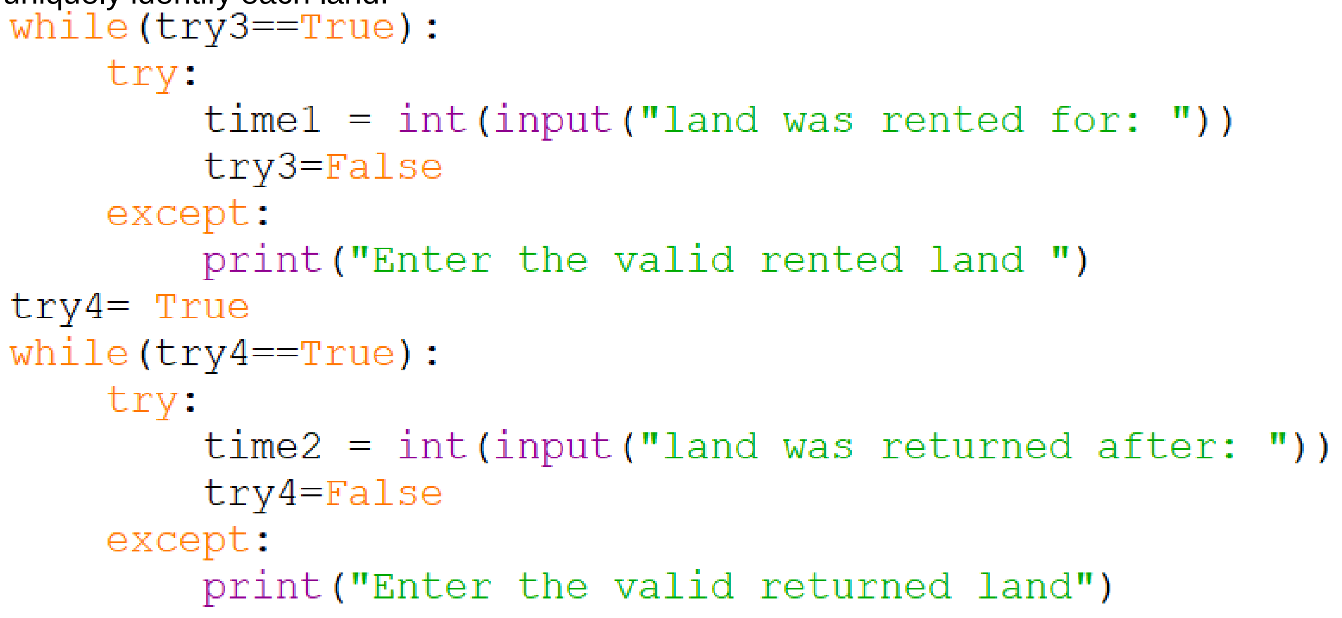
Sets

Sets are an unordered collection of unique elements. Like Lists, Sets are mutable and written within square brackets, but no two values can be the same. Some Set methods include count(), index(), any(), all(), etc (upGrad, 2020).

Example: - unique = {1,2,3,4,5}

Data Structure

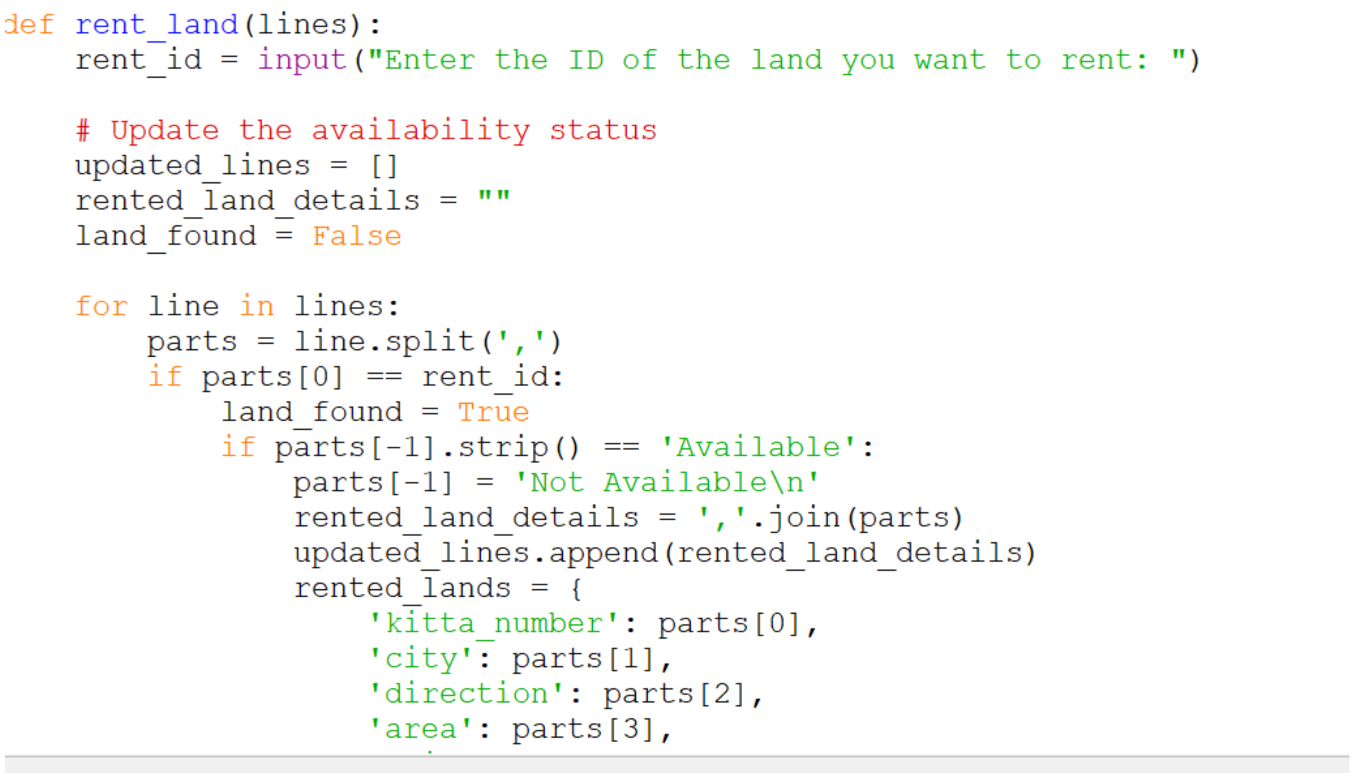
Integers (int): When representing numerical quantities without integer components, integers are used. They are used for attributes such as area, price, duration in months, and kitta number. For example, land and price are the quantitative features of the lands, whereas kitta numbers uniquely identify each land.



```
while(try3==True):
    try:
        time1 = int(input("land was rented for: "))
        try3=False
    except:
        print("Enter the valid rented land ")
try4= True
while(try4==True):
    try:
        time2 = int(input("land was returned after: "))
        try4=False
    except:
        print("Enter the valid returned land")
```

Figure 5.Use of integers

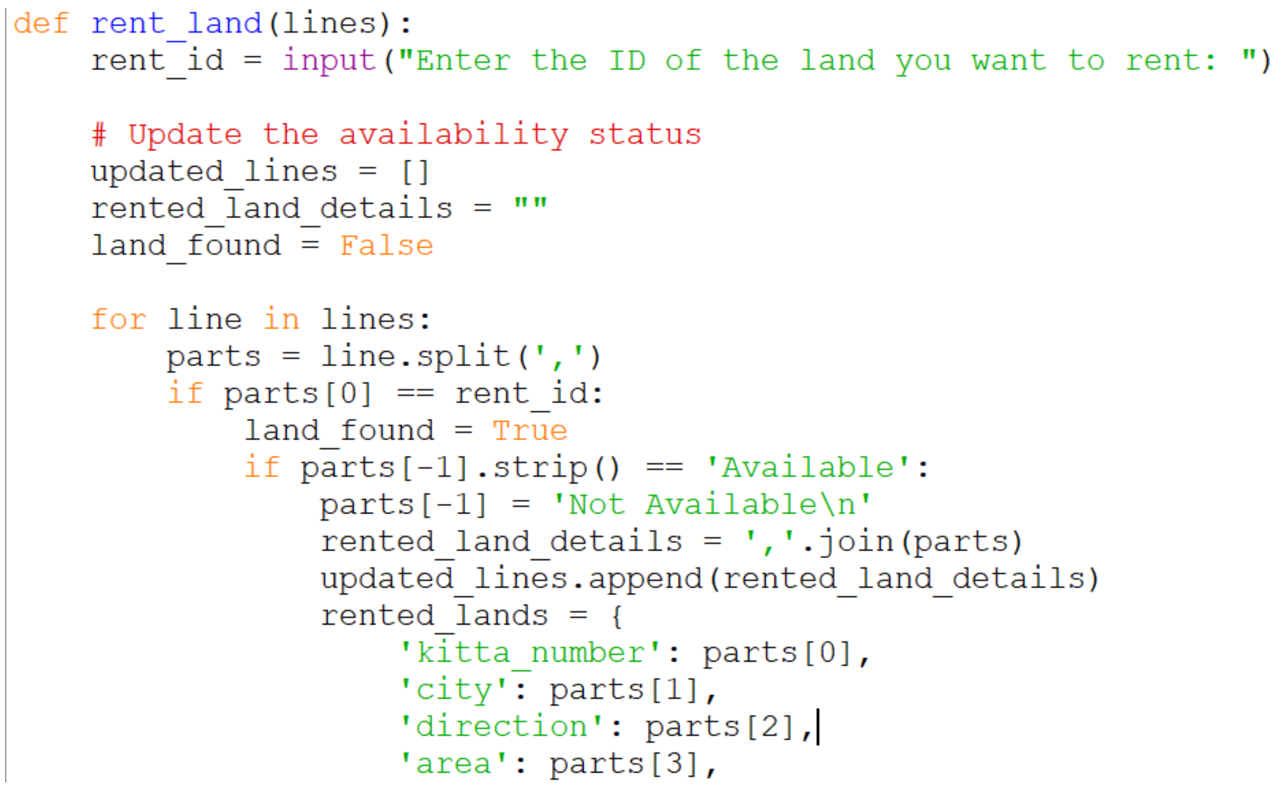
Lists: Lists are sequences that can be changed and contain elements of various data types



```
def rent_land(lines):
    rent_id = input("Enter the ID of the land you want to rent: ")

    # Update the availability status
    updated_lines = []
    rented_land_details = ""
    land_found = False

    for line in lines:
        parts = line.split(',')
        if parts[0] == rent_id:
            land_found = True
            if parts[-1].strip() == 'Available':
                parts[-1] = 'Not Available\n'
                rented_land_details = ','.join(parts)
                updated_lines.append(rented_land_details)
                rented_lands = {
                    'kitta_number': parts[0],
                    'city': parts[1],
                    'direction': parts[2],
                    'area': parts[3],
```

Figure 6: Use of list in renting and returning lands

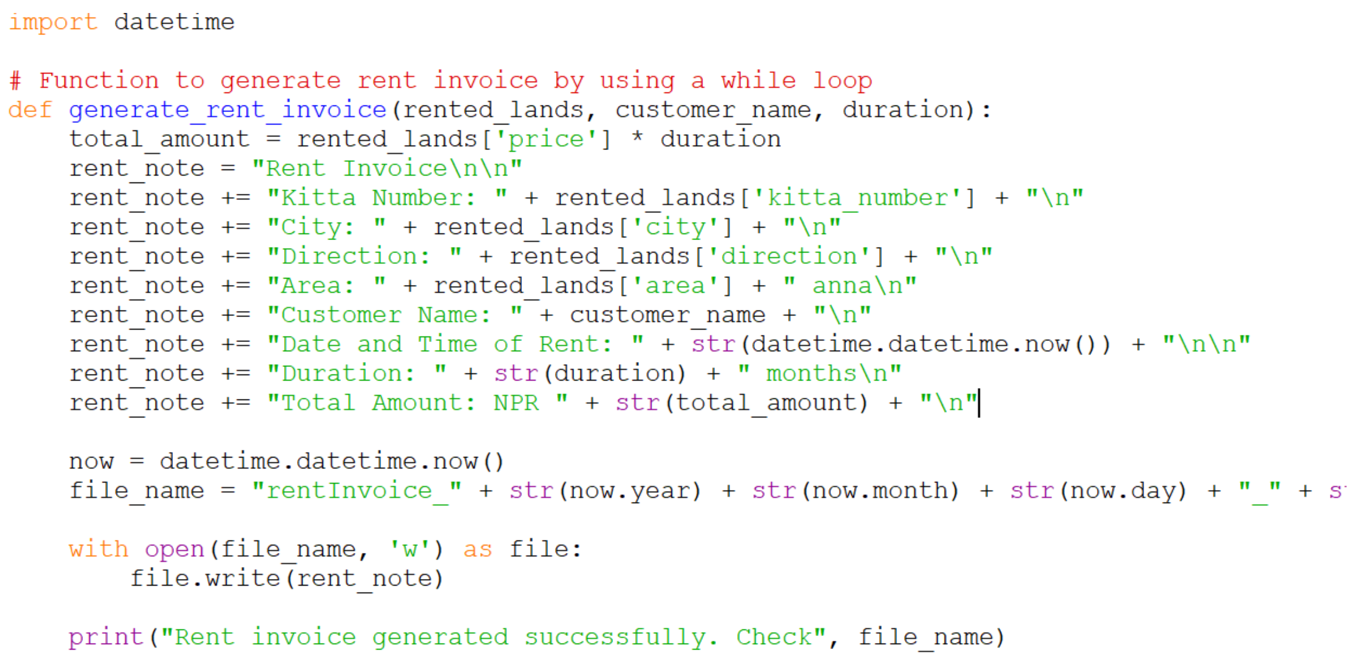
Dictionaries: Dictionaries key-value pair storage is done by modifiable data structures. Code uses dictionaries to represent unique land data. Specific information about each land is stored in a dictionary with keys like "kitta_number," "city," "direction," "area," "price," and "status," and matching values.



```
def rent_land(lines):
    rent_id = input("Enter the ID of the land you want to rent: ")

    # Update the availability status
    updated_lines = []
    rented_land_details = ""
    land_found = False

    for line in lines:
        parts = line.split(',')
        if parts[0] == rent_id:
            land_found = True
            if parts[-1].strip() == 'Available':
                parts[-1] = 'Not Available\n'
                rented_land_details = ','.join(parts)
                updated_lines.append(rented_land_details)
                rented_lands = {
                    'kitta_number': parts[0],
                    'city': parts[1],
                    'direction': parts[2],
                    'area': parts[3],
```

Figure 7: Use of dictionaries

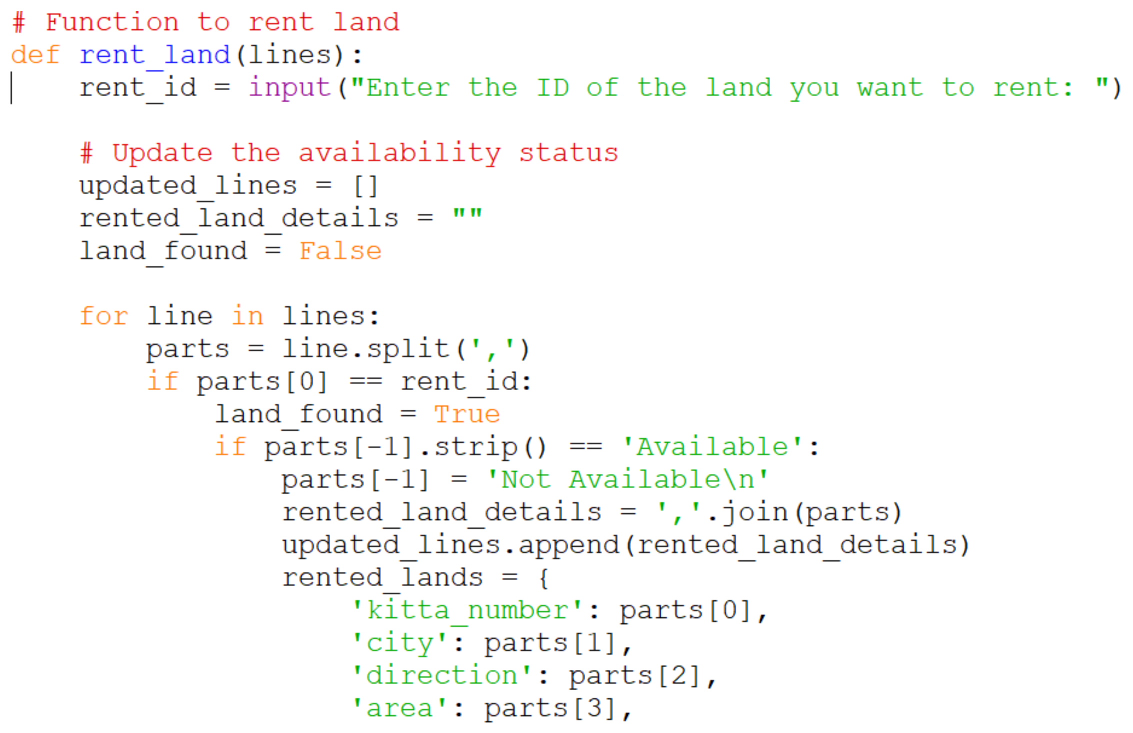
Datetime Objects: Datetime objects describe durations and points in time. In order to accurately track time and calculate rental durations and fines, they are used to record the date and time of renting and returning lands. Datetime objects give me the ability to work with dates and times in my code more effectively.



```python
import datetime

# Function to generate rent invoice by using a while loop
def generate_rent_invoice(rented_lands, customer_name, duration):
    total_amount = rented_lands['price'] * duration
    rent_note = "Rent Invoice\n\n"
    rent_note += "Kitta Number: " + rented_lands['kitta_number'] + "\n"
    rent_note += "City: " + rented_lands['city'] + "\n"
    rent_note += "Direction: " + rented_lands['direction'] + "\n"
    rent_note += "Area: " + rented_lands['area'] + " anna\n"
    rent_note += "Customer Name: " + customer_name + "\n"
    rent_note += "Date and Time of Rent: " + str(datetime.datetime.now()) + "\n\n"
    rent_note += "Duration: " + str(duration) + " months\n"
    rent_note += "Total Amount: NPR " + str(total_amount) + "\n"

    now = datetime.datetime.now()
    file_name = "rentInvoice_" + str(now.year) + str(now.month) + str(now.day) + "_" + s

    with open(file_name, 'w') as file:
        file.write(rent_note)

    print("Rent invoice generated successfully. Check", file_name)
```

Figure 8: Import Time & Date

2D list: 2D list in Python is a common task when working with matrices, tables, or any other structured data. It allows you to extract specific portions of the list, making it easier to manipulate and analyze the data. This tabular format enables efficient organization and manipulation of multiple land details, supporting functionalities such as displaying, selecting, and updating lands within your code.

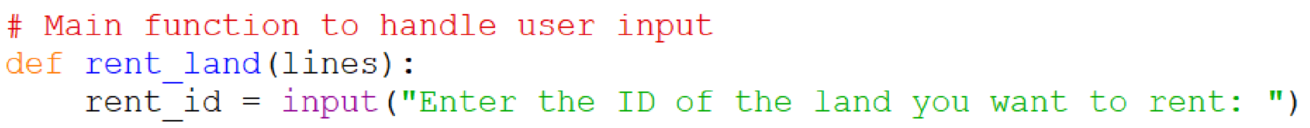


```python
# Function to rent land
def rent_land(lines):
    rent_id = input("Enter the ID of the land you want to rent: ")

    # Update the availability status
    updated_lines = []
    rented_land_details = ""
    land_found = False

    for line in lines:
        parts = line.split(',')
        if parts[0] == rent_id:
            land_found = True
            if parts[-1].strip() == 'Available':
                parts[-1] = 'Not Available\n'
                rented_land_details = ','.join(parts)
                updated_lines.append(rented_land_details)
                rented_lands = {
                    'kitta_number': parts[0],
                    'city': parts[1],
                    'direction': parts[2],
                    'area': parts[3],
```

Figure 9: Use of 2D list

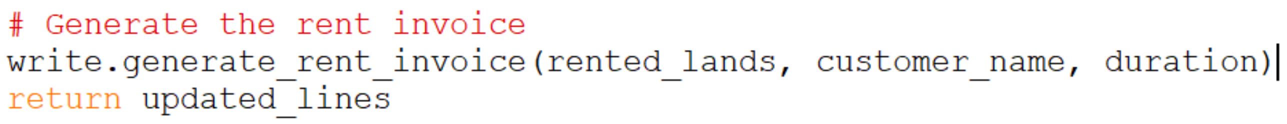
6. Implementation

Modularity refers to the process of dividing a program into more manageable, separate, and compact modules. These modules carry out particular functions, which facilitate simpler comprehension, upkeep, and debugging of the code. Within my code each function covers a specific piece of functionality, encouraging code reuse and making it simpler to test and maintain. Examples of functions include reading and updating land details, handling user input, and producing rent invoices and return invoices. It is clear that the concerns are divided, with various functions taking care of various parts of the program logic.



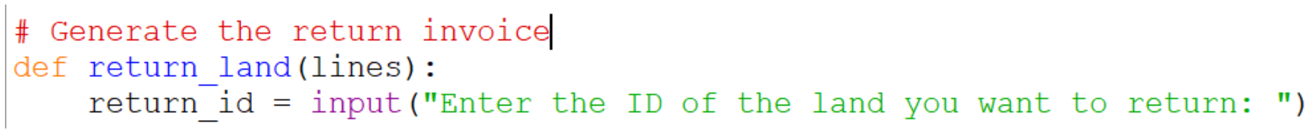
```
# Main function to handle user input
def rent_land(lines):
    rent_id = input("Enter the ID of the land you want to rent: ")
```

Figure 10: Screenshot of main function



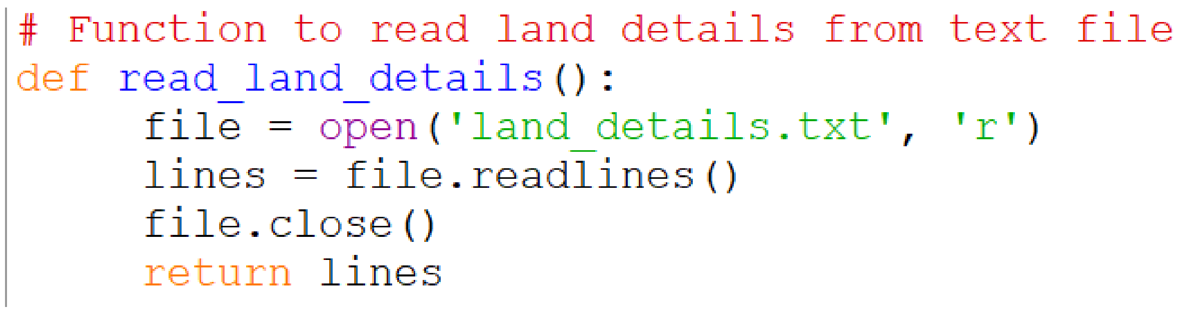
```
# Generate the rent invoice
write.generate_rent_invoice(rented_lands, customer_name, duration)
return updated_lines
```

Figure 11: Screenshot of rent invoice



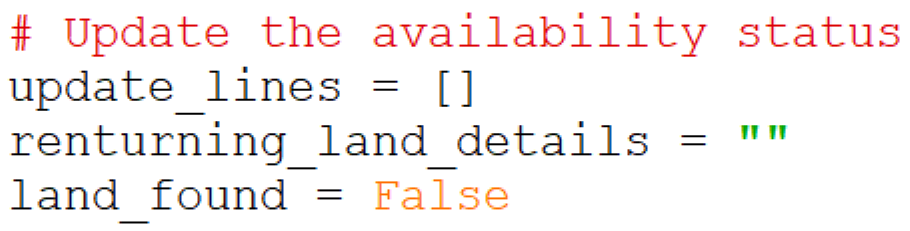
```
# Generate the return invoice
def return_land(lines):
    return_id = input("Enter the ID of the land you want to return: ")
```

Figure 12: Screenshot of return invoice



```
# Function to read land details from text file
def read_land_details():
    file = open('land_details.txt', 'r')
    lines = file.readlines()
    file.close()
    return lines
```

Figure 13: Screenshot of read land details

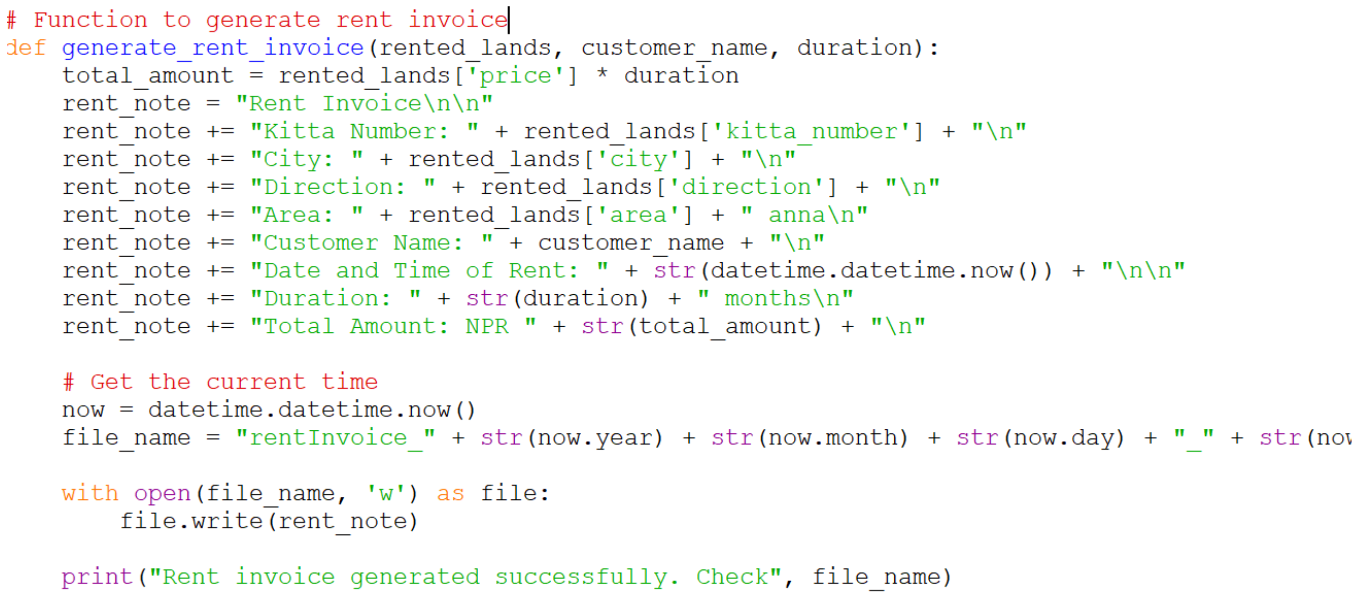


```
# Update the availability status
update_lines = []
renturning_land_details = ""
land_found = False
```

Figure 14: Screenshot of update land detail

7. Programming Style

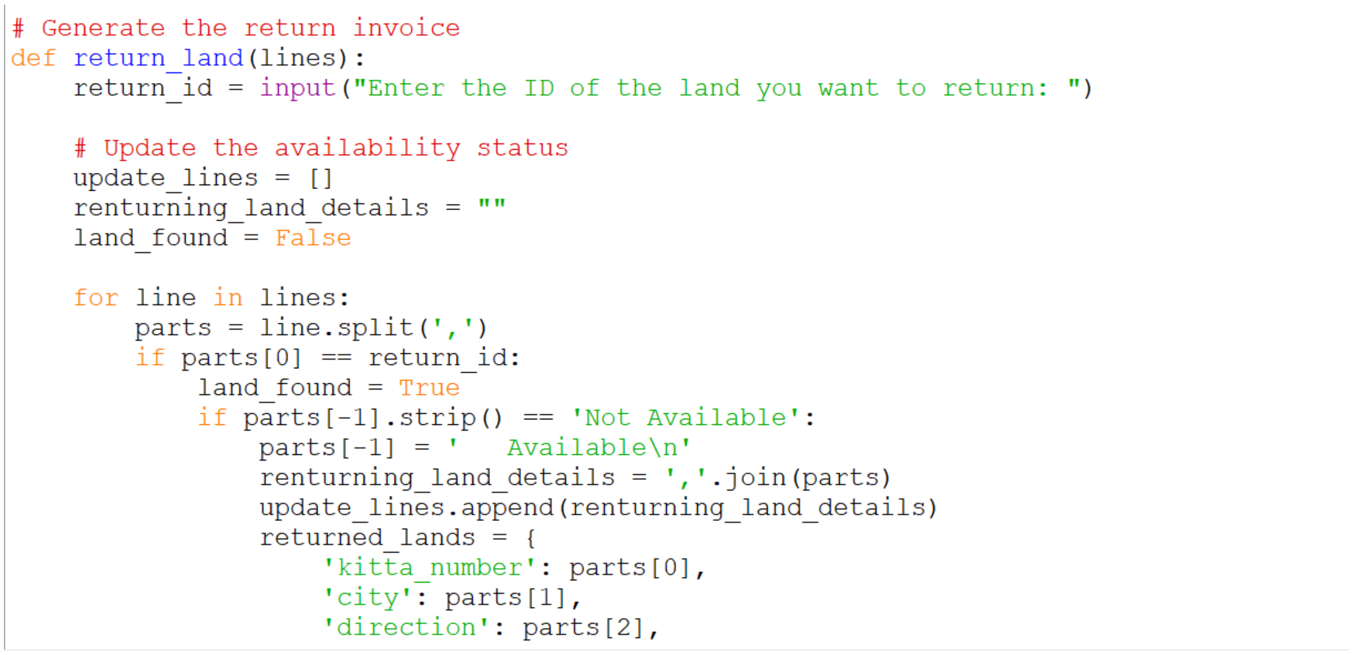
In the code, Users can choose which lands to rent by entering their name and the length of the supposed rental, thanks to the renting functionality. Following an announcement asking users to select among available lands, the program verifies their choices to make sure the selected lands are still accessible for rental. When proper choices are made, the application creates a rent invoice that includes the rented lands, customer name, rent time, duration, and total cost, modifies the chosen lands' status to "Not Available," and stores the data to a file.



```
# Function to generate rent invoice
def generate_rent_invoice(rented_lands, customer_name, duration):
    total_amount = rented_lands['price'] * duration
    rent_note = "Rent Invoice\n\n"
    rent_note += "Kitta Number: " + rented_lands['kitta_number'] + "\n"
    rent_note += "City: " + rented_lands['city'] + "\n"
    rent_note += "Direction: " + rented_lands['direction'] + "\n"
    rent_note += "Area: " + rented_lands['area'] + " anna\n"
    rent_note += "Customer Name: " + customer_name + "\n"
    rent_note += "Date and Time of Rent: " + str(datetime.datetime.now()) + "\n\n"
    rent_note += "Duration: " + str(duration) + " months\n"
    rent_note += "Total Amount: NPR " + str(total_amount) + "\n"

    # Get the current time
    now = datetime.datetime.now()
    file_name = "rentInvoice_" + str(now.year) + str(now.month) + str(now.day) + "_" + str(no

    with open(file_name, 'w') as file:
        file.write(rent_note)

    print("Rent invoice generated successfully. Check", file_name)
```

Figure 15: Screenshot of rent invoice

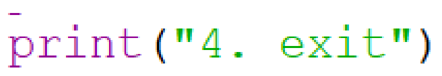
Returning lands, users put their name and the kitta number of the land they want to return to. After confirming the data, the program changes the status of the returned land to "Available." It finds any fines that may be due, creates a return invoice that includes information on the returned land, the customer's name, the time and duration of the return, and the total sum due, including any fines, and files this data.



```
# Generate the return invoice
def return_land(lines):
    return_id = input("Enter the ID of the land you want to return: ")

    # Update the availability status
    update_lines = []
    renturning_land_details = ""
    land_found = False

    for line in lines:
        parts = line.split(',')
        if parts[0] == return_id:
            land_found = True
            if parts[-1].strip() == 'Not Available':
                parts[-1] = '    Available\n'
                renturning_land_details = ','.join(parts)
                update_lines.append(renturning_land_details)
                returned_lands = {
                    'kitta_number': parts[0],
                    'city': parts[1],
                    'direction': parts[2],
```

Figure 16: Screenshot of return invoice

Finally, users can choose to end the program's operation by clicking the exit button; the software will provide an empty exit message that ensures an easy exit.



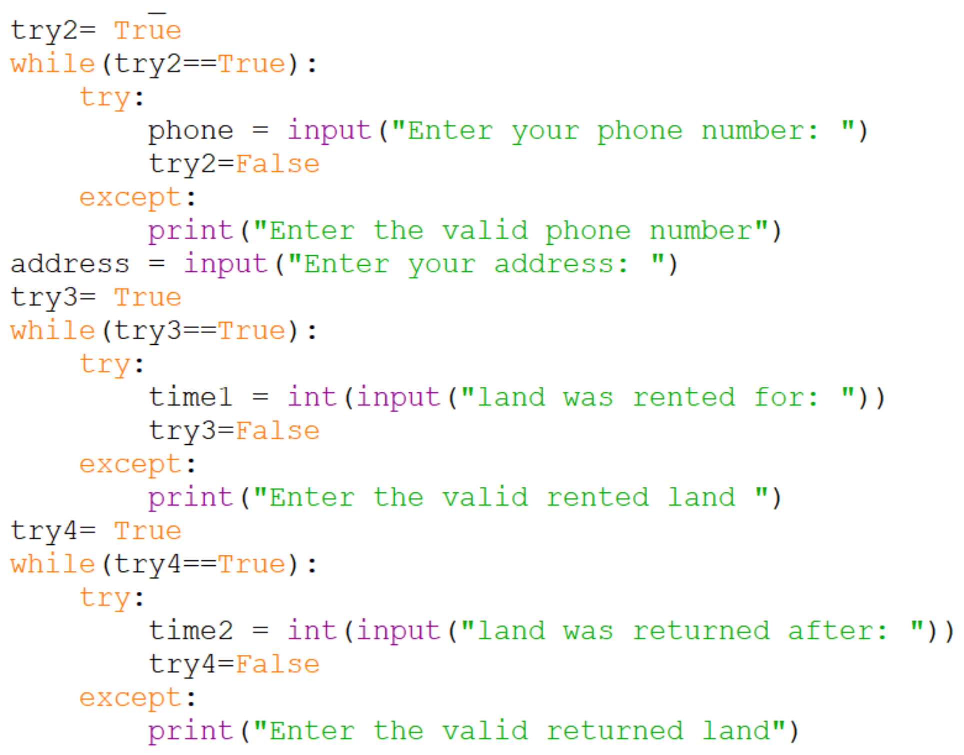
```
print("4. exit")
```

Figure 17: Screenshot of exit system

8. Exception Handling

Exception handling is the process of responding to unwanted or unexpected events when a computer program runs. Exception handling deals with these events to avoid the program or system crashing, and without this process, exceptions would disrupt the normal operation of a program.

User Input Validation: for renting or returning lands, users provide their name, rental period, and kitta number. You've included input validation to make sure the entered values are of the right data type. Try-except blocks with ValueError, for instance, are used to catch situations in which users enter non-integer values for duration or kitta number. By managing these exceptions, you can stop possible crashes brought on by erroneous data types by asking users to provide appropriate inputs.



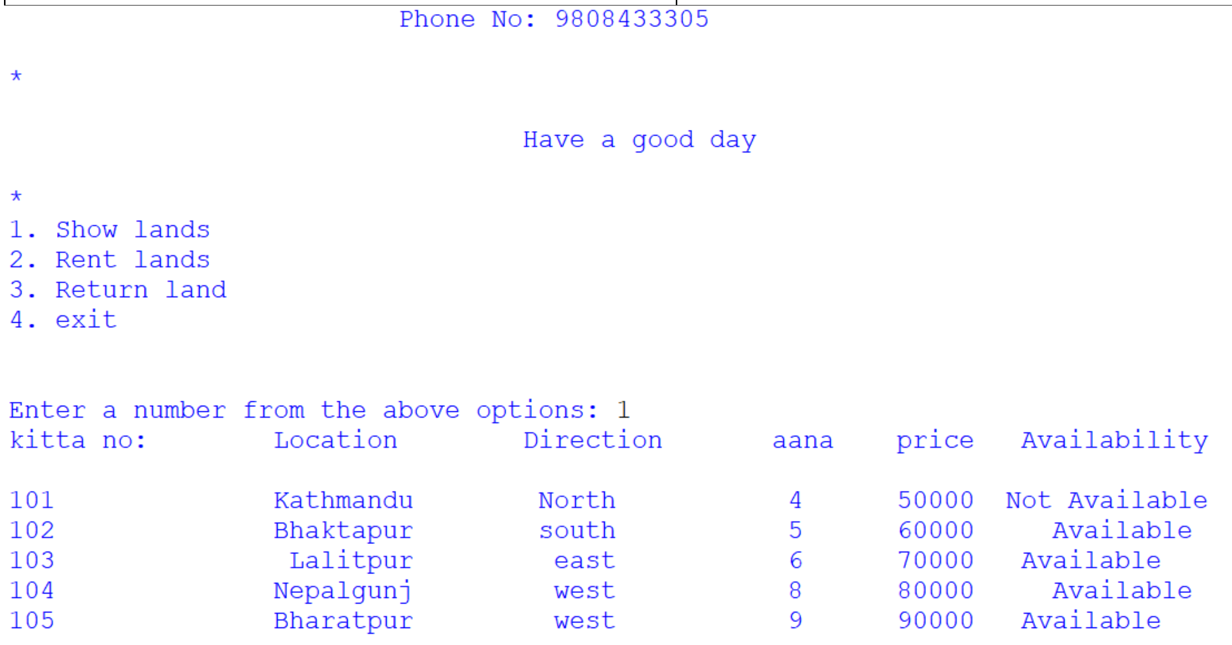
```
try2= True
while(try2==True):
    try:
        phone = input("Enter your phone number: ")
        try2=False
    except:
        print("Enter the valid phone number")
address = input("Enter your address: ")
try3= True
while(try3==True):
    try:
        time1 = int(input("land was rented for: "))
        try3=False
    except:
        print("Enter the valid rented land ")
try4= True
while(try4==True):
    try:
        time2 = int(input("land was returned after: "))
        try4=False
    except:
        print("Enter the valid returned land")
```

Figure 18: User Input Validation try except

9. Testing

1. Table 1: To check the all lands for rent

Test No	9.1
Objective	To check the all lands for rent
Action	The display is called with the following arguments: Press 1 to display all lands Press = 1
Expected Result	To Show all lands
Actual Result	To Show all lands
Conclusion	The test is successful.

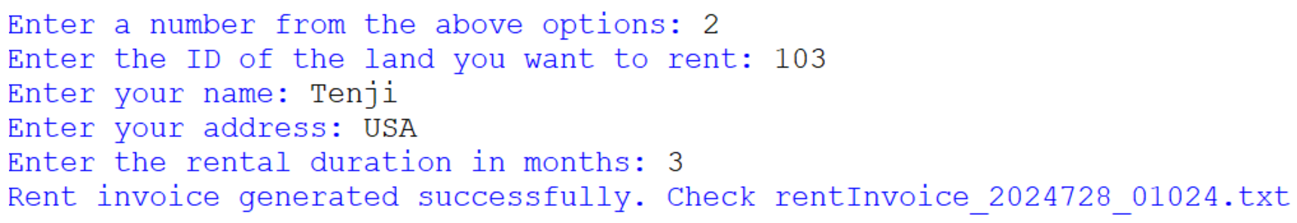


```
                         Phone No: 9808433305

*

                                 Have a good day

*
1. Show lands
2. Rent lands
3. Return land
4. exit


Enter a number from the above options: 1
kitta no:        Location        Direction        aana    price   Availability

101              Kathmandu        North            4      50000  Not Available
102              Bhaktapur        south            5      60000     Available
103               Lalitpur         east            6      70000   Available
104              Nepalgunj         west            8      80000     Available
105              Bharatpur         west            9      90000   Available
```

Figure 19: ScreenShot of all lands

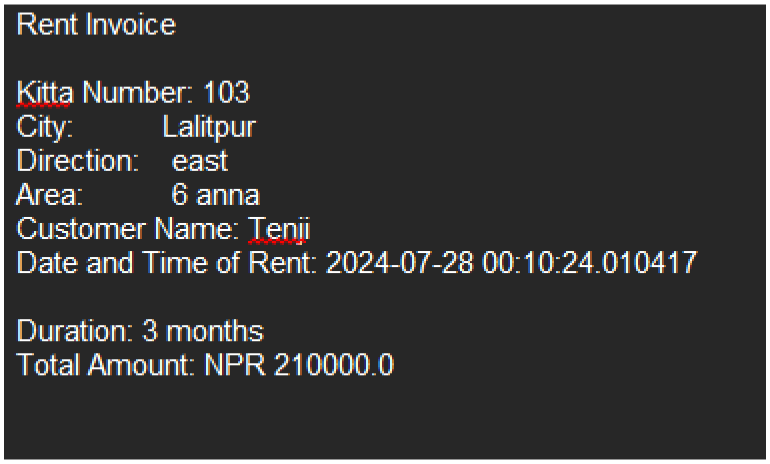
2. Table 2: Generate the rent lands

Test No	9.2
Objective	To check the rent lands
Action	The display is called with the following arguments: Enter your name: Tenji Enter duration of rent: 3 Enter Kitta Numbers of the lands you want to rent: 103
Expected Result	Rent generate successfully
Actual Result	Rent generate successfully
Conclusion	The test is successful.



```
Enter a number from the above options: 2
Enter the ID of the land you want to rent: 103
Enter your name: Tenji
Enter your address: USA
Enter the rental duration in months: 3
Rent invoice generated successfully. Check rentInvoice_2024728_01024.txt
```

Figure 20 : Screenshot of rent land

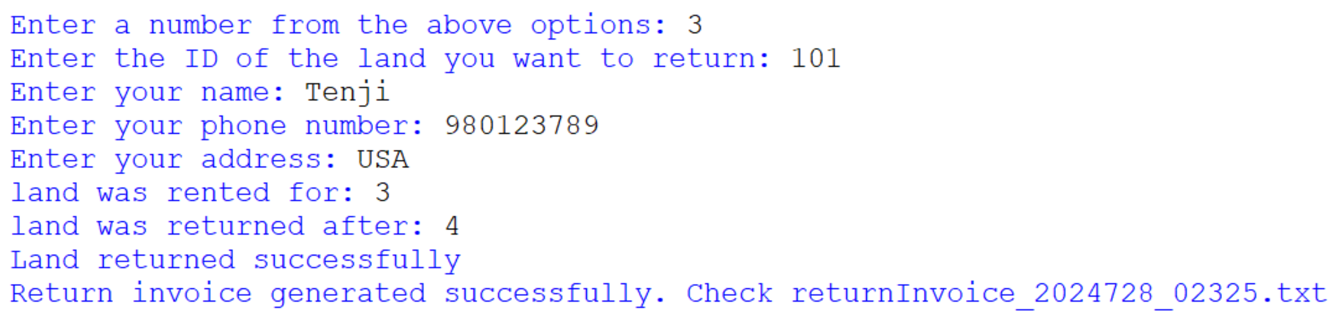


```
Rent Invoice

Kitta Number: 103
City:          Lalitpur
Direction:     east
Area:          6 anna
Customer Name: Tenji
Date and Time of Rent: 2024-07-28 00:10:24.010417

Duration: 3 months
Total Amount: NPR 210000.0
```

Figure 21: Screenshot of renting lands output

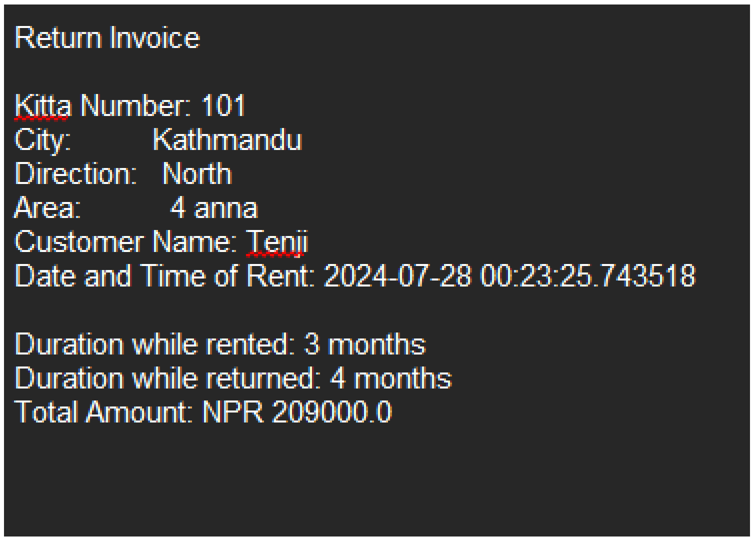
3. Table 3.Generate the return invoice

Test No	9.3
Objective	To check the return invoice
Action	The display is called with the following arguments: Enter Kitta Numbers of the lands you want to return: 101 Enter your name: Tenji Enter the month the land is being returned: 3
Expected Result	Return invoice generate successfully.
Actual Respect	Return invoice generate successfully.
Conclusion	The test is successful.



```
Enter a number from the above options: 3
Enter the ID of the land you want to return: 101
Enter your name: Tenji
Enter your phone number: 980123789
Enter your address: USA
land was rented for: 3
land was returned after: 4
Land returned successfully
Return invoice generated successfully. Check returnInvoice_2024728_02325.txt
```

Figure 22: Screenshot of Return lands



```
Return Invoice

Kitta Number: 101
City:        Kathmandu
Direction:   North
Area:         4 anna
Customer Name: Tenji
Date and Time of Rent: 2024-07-28 00:23:25.743518

Duration while rented: 3 months
Duration while returned: 4 months
Total Amount: NPR 209000.0
```

Figure 23: Screenshot of Return lands output

8. Conclusion

Throughout the project, we have learned the basic of Python programming language very deeply. We have gained main concepts like file handling, exception handling, collection data types and many more. We got so many new ideas while developing an application for an inventory management within the context of real-world scenarios. As an individual undertaking this coursework, we got the opportunity to test my skills, ideas, and understanding of programming skills related to Python programming language.

At first, we faced a lot of problems and many obstacles during the process of developing this application. We could use the two-dimensional list properly in my project. We were so confused where and how to start this coursework. After too much research, dedication, and hard work, we could create the basic interface of my program. My fellow friends and teachers helped us a lot in this beautiful journey. We also got so many syntaxes error while defining own function. Sometimes, we used to forget to give proper indentation while writing the code that irritated me a lot. In other hands, exception handling was just another big obstacle while developing this application. We used to enter string value as input instead of passing integer value that helped to crash my program in the middle of processing of renting or returning equipment.

At last, we were able to complete my project successfully with the help of my fellow friends and teachers who guide me a lot what to do and what not to do while writing the program. Sometimes, we faced syntax errors while implementing the concept of file handling. But at the end, we completed my coursework after giving my full focus, time and making some sacrifices. The concepts and knowledge we used in this course will help me in the future to get placements at big tech-companies like Google, Twitter, and many mores. This coursework also helped me to increase problem solving skill, creativity and thinking capacity of new ideas. Overall, we enjoyed this journey in this journey while developing this application.

9. Bibliography

CFI, 2021. *Python Data Structures - Overview, Types, Examples.* [Online]
Available at: https://corporatefinanceinstitute.com/resources/knowledge/other/python-data-structures/
[Accessed 20 August 2022].
CodeSansar, 2022. *Fundamental or Primitive Data Types in Python.* [Online]
Available at: https://www.codesansar.com/python-programming/fundamental-or-primitive-data-types.htm#:~:text=Python%20programming%20language%20has%20four,%2C%20floats%2C%20booleans%20and%20strings.
[Accessed 20 August 2022].
EDUCBA, 2022. *Complete Guide to Python pseudocode - eduCBA.* [Online]
Available at: https://www.educba.com/python-pseudocode/
[Accessed 18 August 2022].
javatpoint, 2022. *Python List.* [Online]
Available at: https://www.javatpoint.com/python-lists
[Accessed 18 August 2022].
lucidchart, 2022. *Flowchart Symbols and Notation.* [Online]
Available at: https://www.lucidchart.com/pages/flowchart-symbols-meaning-explained
[Accessed 16 August 2022].
programiz, 2022. *Learn Data Structures and Algorithms.* [Online]
Available at: https://www.programiz.com/dsa/data-structure-types
[Accessed 16 August 2022].
upGrad, 2020. *Data Structures & Algorithm in Python: Everything You Need ....* [Online]
Available at: https://www.upgrad.com/blog/data-structures-algorithm-in-python/#:~:text=Algebra%20for%20Analysis-,What%20are%20algorithms%20in%20Python%3F,guide%20the%20writing%20of%20algorithms.
[Accessed 23 August 2022].
upGrad, 2020. *Data Structures & Alogorithm in Python:Everything You Need...* [Online]
Available at: https://www.upgrad.com/blog/data-structures-algorithm-in-python/#:~:text=Algebra%20for%20Analysis-,What%20are%20algorithms%20in%20Python%3F,guide%20the%20writing%20of%20algorithms.
[Accessed 23 August 2022].

10. Appendix

```
#read.py
# Function to read land details from text file
def land_read_details():
     file = open('land_details.txt', 'r')
     lines = file.readlines()
     file.close()
     return lines


# Function to display land details in a table format
def show_land_details(lines):
    file = open('Land_details.txt', 'r')
    lines = file.readlines()
    file.close()

    # Print the table header
    header = lines[0].split(',')
    for column in header:
        if column[-1] == '\n':
            column = column[:-1]
        print(column, end='\t')
    print()

    # Print the table rows
    for line in lines[1:]:
        row = line.split(',')
        for cell in row:
            if cell[-1] == '\n':
                cell = cell[:-1]
            print(cell, end='\t')
        print()



# operation.py

import read
import write
#Function to generate rent land
def rent_land(lines):
    rent_id = input("Enter the ID of the land you want to rent: ")

    # Update the availability status
    updated_lines = []
    rented_land_details = ""
```

```
land_found = False

for line in lines:
    parts = line.split(',')
    if parts[0] == rent_id:
        land_found = True
        if parts[-1].strip() == 'Available':
            parts[-1] = 'Not Available\n'
            rented_land_details = ','.join(parts)
            updated_lines.append(rented_land_details)
            rented_lands = {
                'kitta_number': parts[0],
                'city': parts[1],
                'direction': parts[2],
                'area': parts[3],
                'price': float(parts[4])
            }
        else:
            print("The selected land is already rented.")
            return lines
    else:
        updated_lines.append(line.strip() + '\n')

if not land_found:
    print("The land ID entered does not exist.")
    return lines

#updated lines
with open('land_details.txt', 'w') as file:
    for line in updated_lines:
        file.write(line)

# Get user details
customer_name = input("Enter your name: ")
address = input("Enter your address: ")
try1= True
while(try1==True):
    try:
        duration = int(input("Enter the rental duration in months: "))
        try1=False
    except:
        print("Enter the valid duration")



write.generate_rent_invoice(rented_lands, customer_name, duration)
```

```
    return updated_lines


# Generate the return invoice
def return_land(lines):
    return_id = input("Enter the ID of the land you want to return: ")

    # Update the availability status
    update_lines = []
    renturning_land_details = ""
    land_found = False

    for line in lines:
        parts = line.split(',')
        if parts[0] == return_id:
            land_found = True
            if parts[-1].strip() == 'Not Available':
                parts[-1] = '   Available\n'
                renturning_land_details = ','.join(parts)
                update_lines.append(renturning_land_details)
                returned_lands = {
                    'kitta_number': parts[0],
                    'city': parts[1],
                    'direction': parts[2],
                    'area': parts[3],
                    'price': float(parts[4])
                }
            else:
                print("The selected land is not rented.")
                return lines
        else:
            update_lines.append(line.strip() + '\n')

    if not land_found:
        print("The land ID entered does not exist.")
        return lines

    with open('land_details.txt', 'w') as file:
        for line in update_lines:
            file.write(line)

        #Fine if land is returned late
    customer_name = input("Enter your name: ")
    try2= True
    while(try2==True):
        try:
            phone = input("Enter your phone number: ")
            try2=False
        except:
            print("Enter the valid phone number")
    address = input("Enter your address: ")
    try3= True
    while(try3==True):
        try:
            time1 = int(input("land was rented for: "))
            try3=False
        except:
            print("Enter the valid rented land ")
    try4= True
    while(try4==True):
        try:
            time2 = int(input("land was returned after: "))
            try4=False
        except:
            print("Enter the valid returned land")

    if time2 > time1:
        total_time = time2 - time1
        fine = int(parts[-2]) * total_time * 0.1

    else:
        fine = 0
    total_price = returned_lands['price'] * time2 + fine



    print("Land returned successfully")
    write.generate_return_invoice(customer_name, phone, address, time1, time2,
returned_lands, fine, total_price )
    return update_lines




import datetime

# Function to generate rent invoice
def generate_rent_invoice(rented_lands, customer_name, duration):
    total_amount = rented_lands['price'] * duration
    rent_note = "Rent Invoice\n\n"
    rent_note += "Kitta Number: " + rented_lands['kitta_number'] + "\n"
    rent_note += "City: " + rented_lands['city'] + "\n"
    rent_note += "Direction: " + rented_lands['direction'] + "\n"
    rent_note += "Area: " + rented_lands['area'] + " anna\n"
    rent_note += "Customer Name: " + customer_name + "\n"
    rent_note += "Date and Time of Rent: " + str(datetime.datetime.now()) + "\n\n"
    rent_note += "Duration: " + str(duration) + " months\n"
    rent_note += "Total Amount: NPR " + str(total_amount) + "\n"

    # Get the current time
    now = datetime.datetime.now()
    file_name = "rentInvoice_" + str(now.year) + str(now.month) + str(now.day) + "_" +
str(now.hour) + str(now.minute) + str(now.second) + ".txt"

    with open(file_name, 'w') as file:
        file.write(rent_note)

    print("Rent invoice generated successfully. Check", file_name)

# Function to generate return invoice
def generate_return_invoice(customer_name, phone, address,time1, time2,
returned_lands, fine, total_price ):
    return_note = "Rent Invoice\n\n"
    return_note += "Kitta Number: " + returned_lands['kitta_number'] + "\n"
    return_note += "City: " + returned_lands['city'] + "\n"
    return_note += "Direction: " + returned_lands['direction'] + "\n"
    return_note += "Area: " + returned_lands['area'] + " anna\n"
    return_note += "Customer Name: " + customer_name + "\n"
    return_note += "Date and Time of Rent: " + str(datetime.datetime.now()) + "\n\n"
    return_note += "Duration while rented: " + str(time1) + " months\n"
    return_note += "Duration while returned: " + str(time2) + " months\n"
    return_note += "Total Amount: NPR " + str(total_price) + "\n"

    # Get the current time
    now = datetime.datetime.now()
    file_name = "returnInvoice_" + str(now.year) + str(now.month) + str(now.day) + "_" +
str(now.hour) + str(now.minute) + str(now.second) + ".txt"

    with open(file_name, 'w') as file:
        file.write(return_note)

    print("Return invoice generated successfully. Check", file_name)




#main.py
```

```
import read
import write
import operation

# Main program to handle user input
def main():
    print("\n\t\t\t Techno Property Nepal")
    print("\n\t\t\t Kamalpokhari, Kathmandu")
    print("\n\t\t\t Phone No: 9808433305")
    print("\n*")
    print("\n\t\t\t\t Have a good day")
    print("\n*")

    print("1. Show lands")
    print("2. Rent lands")
    print("3. Return land")
    print("4. exit")

    lines = read.land_read_details()

    while True:
        choice = input("\n\nEnter a number from the above options: ")
        if choice == "1":
            read.show_land_details(lines)
        elif choice == "2":
            lines = operation.rent_land(lines)
        elif choice == "3":
            lines = operation.return_land(lines)
        elif choice == "4":
            return

        else:
            print("Invalid input, please try again.")


main()
```