

**Module Code & Module Title:**

**CS4051NT Fundamentals of Computing**

**Assessment Weightage & Type:**

**60% Individual Coursework**

**Year and Semester:**

**2024 Spring**

**Student Name: Anmol Poudyal**

**London Met ID: 23049190**

**College ID: np05cp4a230006@iic.edu.np**

**Assignment Due Date: 07 May**

**Word Count: 5163**

*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 mark of zero will be awarded.*

Table of Contents

[1 Introduction 3](#_Toc165714754)

[1.1 Python 3](#_Toc165714755)

[1.2 Introduction to Project 4](#_Toc165714756)

[1.3 Intrduction to IDLE 4](#_Toc165714757)

[1.4 Introduction to Microsoft Word 5](#_Toc165714758)

[1.5 Introduction to Draw.io 5](#_Toc165714759)

[1.6 Goal and objective 6](#_Toc165714760)

[2 Discussion and analysis 6](#_Toc165714761)

[2.1 Algorithm 6](#_Toc165714762)

[2.2 Pseudocode 9](#_Toc165714763)

[3 Flowchart 14](#_Toc165714764)

[4 Data Structure 16](#_Toc165714765)

[5 Program 18](#_Toc165714766)

[6 Testing 22](#_Toc165714767)

[6.1 Test 1 Show implementation of try, except 22](#_Toc165714768)

[6.2 Test 2 Selection rent and return of lands 23](#_Toc165714769)

[6.3 Test 3 File generation of renting of land(s) (Renting multiple land(s)) 24](#_Toc165714770)

[6.4 Test 4 File generation of returning process of land(s) (Returning multiple land(s)) 27](#_Toc165714771)

[7 Conclusion 35](#_Toc165714772)

[8 References 35](#_Toc165714773)

[9 Appendix 35](#_Toc165714774)

**Table of figures:**

[Figure 1:Pyhon logo 6](#_Toc165717050)

[Figure 2: IDLE logo 7](#_Toc165717051)

[Figure 3: MSWord logo 8](#_Toc165717052)

[Figure 4: Draw.io logo 8](#_Toc165717053)

[Figure 5: flowchart of the program 17](#_Toc165717054)

[Figure 6: Breaking down flow chart for more understandable from 18](#_Toc165717055)

[Figure 7: String example screenshot 20](#_Toc165717056)

[Figure 8: Integer example screenshot 20](#_Toc165717057)

[Figure 9: List example screen shot 21](#_Toc165717058)

[Figure 10: rent of the land screen shot 23](#_Toc165717059)

[Figure 11: return of the land 23](#_Toc165717060)

[Figure 12: fine calculation and cost calculation 24](#_Toc165717061)

[Figure 13: generation of text file for renting land 24](#_Toc165717062)

[Figure 14: generating text file for the return of land 24](#_Toc165717063)

[Figure 15: termination of program after selecting option 25](#_Toc165717064)

[Figure 16: evidence for the implimentation of try , except 25](#_Toc165717065)

[Figure 17: choosing to select rent 26](#_Toc165717066)

[Figure 18:negative input Kitta no 26](#_Toc165717067)

[Figure 19: output after entering negative kitta\_no number 26](#_Toc165717068)

[Figure 20: existing kitta\_number's 27](#_Toc165717069)

[Figure 21: entering non existing value of kitta\_no 27](#_Toc165717070)

[Figure 22: Message after entering non existing value 27](#_Toc165717071)

[Figure 23: choosing option to rent land 28](#_Toc165717072)

[Figure 24: inserting inputs for renting kitta\_no 110 28](#_Toc165717073)

[Figure 25: shell output for renting the first land 28](#_Toc165717074)

[Figure 26:Output for the fist land rented in text file 28](#_Toc165717075)

[Figure 27: inputs for second Land with kitta\_no 114 29](#_Toc165717076)

[Figure 28: output of multiple land rented in shell 29](#_Toc165717077)

[Figure 29 Output for multiple renting in .txt file 30](#_Toc165717078)

[Figure 30: choosing returning option 31](#_Toc165717079)

[Figure 31: entering inputs for returning land with kitta no 103 31](#_Toc165717080)

[Figure 32: shell output as invoice for returning land 103(first return) 31](#_Toc165717081)

[Figure 33: output for the first land returned by same customer 32](#_Toc165717082)

[Figure 34: inputs for returning second land of kitta\_no 107 by same customer 32](#_Toc165717083)

[Figure 35: invoice printed in shell for returning 2 lands by same person 32](#_Toc165717084)

[Figure 36: output invoice of returning multiple land in .txt file 33](#_Toc165717085)

[Figure 37: Land data info in shell 34](#_Toc165717086)

[Figure 38: land field data in .txt file 34](#_Toc165717087)

[Figure 39: choosing to rent this(Before renting) 35](#_Toc165717088)

[Figure 40: choosing rent option 35](#_Toc165717089)

[Figure 41: successful rent 35](#_Toc165717090)

[Figure 42 after rent was successful status changed in shell 36](#_Toc165717091)

[Figure 43: status changed in .txt file after successful rent 36](#_Toc165717092)

[Figure 44: before returningof land in shell 36](#_Toc165717093)

[Figure 45: before returning of land in .txt file 36](#_Toc165717094)

[Figure 46: successful renturning of kitta no 114 37](#_Toc165717095)

[Figure 47: change in status of land in shell after successful returning 37](#_Toc165717096)

[Figure 48: changing status of land in file after successful returning 37](#_Toc165717097)

**Table of tables**

[Table 1: test 1 showing implementation of try except 25](#_Toc165717098)

[Table 2 :Test 2 Selection rent and return of lands 26](#_Toc165717099)

[Table 3: test-3 File generation of renting of land(s) (Renting multiple land(s)) 27](#_Toc165717100)

[Table 4: test-4 File generation of returning process of land(s) (Returning multiple land(s)) 30](#_Toc165717101)

[Table 5 Test-5 Show the update in stock of land(s) 33](#_Toc165717102)

# Introduction

## Python

Python is a high-level programming language developed by Guido Van Rossum in 1991. It’s a programming language which is dynamic typed and dynamic binded, along with its high-level built-in data structures like dictonaries and list, make it one of the widely used language for Application Development areas. It is also used as a scripting language to join existing components alltogether.

Python has readability, which lowers software maintenance costs. Python's support for packages and modules promotes code reuse and program modularity. The large standard library and the Python interpreter are freely distributable and accessible for free on all major platforms in source or binary form. (Foundation, 2024)

Python now is basically used for data analysis, machine learning, web developnment, automation or writing script, software testingetc one of the main resons for its popularity is its simple syntax and beginer friendly as it supports and encourage entry-level coders. (Coursera, apr 4 2024)

A logo of a python company

Description automatically generated

Figure 1:Pyhon logo

## Introduction to Project

This coursework was created using Python v2024.4.1 . the main goal of this project was

to develop a land rental system which allows a user to buy the land or after renting the land he/she could return the land back with proper invoice generated. When invoice is generated the programme must calculate cost during rental invoice generation as well as calculate amount with fine if the rented month is exceeded. In the invoice details including name of the customer, address, numbers of kitta he rented also must be returned. All the information regarding land was asked to store in one file and during a successful rent and return the status of the land i.e “Available” and “Not-Available” must be changed.

## Intrduction to IDLE

Ideal or simply(integraded developnment and learning environment) is a python integrated development environment included in the standard release of Python.

Its one of the user friendly IDE and consists of intelligent indentation feature which makes easier for the programmers to wirte their code smoothly.

A yellow and blue logo

Description automatically generated

Figure 2: IDLE logo

## Introduction to Microsoft Word

Microsoft word or msword is one of most used word processor developed by Microsoft in 1983. In this course work I have used microsoft word for the documentation purpose.

A blue logo with white text

Description automatically generated

Figure 3: MSWord logo

## Introduction to Draw.io

Draw.io is free online diagram software that is used for drawing flowchart and other diagrams. Among the many functions offered by the web-based diagramming application draw.io are flowcharting, network diagrams, and UML diagrams. Diagrams can be accessed and shared with ease thanks to its integration with well-known cloud services like Dropbox, OneDrive, and Google Drive. (UpGuard, 2024)



Figure 4: Draw.io logo

## Goal and objective

Goal: The main goal of this project was to create userfriendly environment for the users to rent, return and view Lands that are either available or not available. The main goal was to make the renting and returing process easier while also providing quick information of the land at the same time.Also it was main goaled to maintain communication between the user and the customer.

Objective :

The main objective of the Project was to gain sound knowledege of

1. Concept of modular programming

2. Uses of function and parameters

3. Different data Structure such as list

4. File handeling

5. Data manipulation in loops as well as syntax

# Discussion and analysis

## Algorithm

An algorithm is a step wise procedure to solve a problem. In hardware- or software-based routines, algorithms function as a precise set of instructions that carry out predetermined operations one after the other. (Alexander S. Gillis, n.d.)

All branches of information technology heavily rely on algorithms. An algorithm is typically used in mathematics, computer science, and computer programming to describe a brief process that resolves a recurring issue. Algorithms are essential to automated systems because they provide as guidelines for processing data. Below the algorithm for building land rental system is done.

**For the program:**

Step-1: Display store menu with options for viewing available lands, renting a land, returning a rented land, and exiting the system.

Step-2: Ask the user to select an option.

Step-3: If the user selects "1", call the function to view available lands.

Step-3.1: Open file land.txt in read mode.

Step-3.2: Read each line in the file one by one

Step-3.3: From each line, check if the substring contains Available

Step-3.4: If found print the line

Step-3.5: if not found repeat step 3.3 and 3.4

Step-3.6: close the file ounce all the lines are read

Step-4: If the user selects "2", call the function to rent a land.

Step-4.1: Take kitta no input form the user.

Step-4.2: Check if the land is rented or not.

Step-4.2.1: if land is rented print “Land is not available”.

Step-4.3: Take rented month input from the user.

Step-4.4: Take name from the user.

Step-4.5: calculate cost using the amount of the land and the given months.

Step-4.6: Open the "Land\_field.txt" file in read mode.

Step-4.7: Iterate through each line in the file.

Step-4.8: Split each line into its components using comma and store in list.

Step-4.9: Check if the input kitta number matches any of the kitta no in a list.

Step-4.10: If extract the city name, direction, and area of the land from the

corresponding fields in the line.

Step-4.11: Call the generate receipt function.

Step-4.11.1: receive the parameters kitta no, rented months, customer.

Name, cost, name of city, direction and area.

Step-4.11.2: open a file with customer name in append mode.

Step-4.11.3: Write details of the land with cost in the file.

Step-4.12: call the function write.rent to change the status of the land to not

available.

Step-4.13: End the function.

Step-5: If the user selects "3", call the function to return a rented land.

Step-5.1: take input kitta no from the user.

Step-5.2: Check if the land is rented or not

Step-5.2.1: if not rented print “not rented yet”

Step-5.3: Take input rented month from the user.

Step-5.4: Take input returned month form the user.

Step-5.5: Take input name from the user.

Step-5.6: Calculate cost using the rent amount from the land details and the month

of rent

Step-5.7: Calculate fine amount using the rented month and returned month

Step-5.8: Open "Land\_field.txt" file in read mode.

Step-5.9: Iterate through each line in the file.

Step-5.10: Check if the kitta number entered by the user matches any kitta number

in the file.

Step-5.11: if found Extract the city name, direction, and area of the land from the

corresponding fields in the line

Step-5.12: call the return invoice function.

Step-5.12.1: Receive the parameter city name, direction, cost, fine, name

, rented month returned month

Step-5.12.2: open a file with customer name in append mode.

Step-5.12.3: Write details of the land with costs in the file.

Step-5.12.4: Change the land status to available

Step-5.12.5: Close the file

Step-5.13: End the function

Step-6: If the user selects "4", exit the system.

Step-7: If the user selects any other option, display a message indicating an invalid choice and prompt the user again.

Step-8: After each operation, ask the user if they want to perform another operation.

Step-9: If the user chooses to perform another operation, repeat steps 2-7. If not, end the program.

## Pseudocode

A pseudocode is the description of algorithm with the explanation of the system in the mixture of both programming(technical words), formal as well as informal language. It helps a programmer to build a specific program more easily and efficiently.

For the main function

**FUNCTION** main():

**LOOP** indefinitely:

**DISPLAY** "Welcome to Land Management System"

**DISPLAY** "1. View Available Lands"

**DISPLAY** "2. Rent a Land"

**DISPLAY** "3. Return a Rented Land"

**DISPLAY** "4. Exit"

**INPUT** (user\_choice )

**IF** user\_choice is empty or not a single digit number:

**DISPLAY** "Invalid input. Please enter a single digit number."

**CONTINUE** **LOOP**

**CONVERT** user\_choice to integer

**IF** user\_choice is 1:

**CALL** function available\_lands()

**ELSE IF** user\_choice is 2:

**CALL** function to rent a land

**ELSE IF** user\_choice is 3:

**CALL** function to return a rented land

**ELSE** **IF** user\_choice is 4:

**EXIT** **LOOP**

**ELSE:**

**DISPLAY** "Invalid choice. Please enter a number between 1 and 4.

**DISPLAY** "Do you want to perform another operation?"

**DISPLAY** "1. Yes"

**DISPLAY** "2. No"

**INPUT** user\_choice

**IF** user\_choice is "2":

**EXIT** **LOOP**

**For function is\_land\_available**

**FUNCTION** is\_land\_available(kitta\_no):

**OPEN** "Land\_field.txt" for reading as file.

**FOR** each line in file:

**SPLIT** line by commas into a list of values

**IF** kitta\_no is in the list of values AND the last element is "Available":

**CLOSE** the file.

**RETURN** True

**CLOSE** the file.

**RETURN** False

**For function is\_land\_rented**

**FUNCTION** is\_land\_rented(kitta\_no):

**OPEN** "Land\_field.txt" for reading as file

**FOR** each line in file:

**SPLIT** line by commas(,) into a list of values

**IF** kitta\_no is in the list of values AND the last element in the value is "Not-Available":

**CLOSE** the file

**RETURN** True

**CLOSE** the file

**RETURN** False

**For function calculate\_cost**

**FUNCTION** calculate\_cost(kitta\_no, months):

**OPEN** "Land\_field.txt" for reading as file

**FOR** each line in file:

**SPLIT** line by commas into a list of values

**IF** kitta\_no is in the list of values:

cost\_per\_month = **CONVERT** fourth element of the list to integer

total\_cost = cost\_per\_month \* months

**CLOSE** the file

**RETURN** total\_cost

**CLOSE** the file

**RETURN** None

**For Function Calculate\_fine**

**FUNCTION calculate\_fine(rented\_month, returned\_month):**

fine = 0

months\_exceeded = returned\_month - rented\_month

**IF months\_exceeded > 0 THEN**

fine = months\_exceeded \* 5000

**RETURN** fine

**For Function Rent\_land()**

**Function** input kitta no

**If** is\_land\_rented in kitta no true

**Display “**Land is not available”

**Return**

**Input** rent\_month

**Input** name

**Call** calculate\_cost function to calculate cost

**Open** the file "Land\_field.txt" in read mode and assign it to the variable 'file'

**LOOP** To read each line file

**Split** the line by comma and strip any leading or trailing whitespaces

**If** the 'kitta\_no' is found in the splitted data:

**Assign** the second element of the splitted data to 'name\_of\_city'

**Assign** the third element of the splitted data to 'direction'

**Assign** the fourth element of the splitted data to 'area'

**Generate** a receipt using the 'generate\_receipt' function with parameters:

**Call** the 'write.rent\_land' function with 'kitta\_no' as a parameter

**Open** the file with the name '<name>.txt' in read mode and assign the value

**Print** the content of the file

**Exit** the loop

**For** **function return\_land**

**Input** kitta\_no.

**If** not is\_land\_rented(kitta\_no):

**DISPLAY** land is not rented yet

**RETURN**

**INPUT** rented month

**INPUT** returned month

**INPUT** name

**Calculate** fine call the calculate\_fine function

**Calculate** cost call the calculate\_cost function

**Open** the file "Land\_field.txt" in read mode and assign it to the variable 'file'

**LOOP** To read each line file

**Split** the line by comma and strip any leading or trailing whitespaces

**If** the 'kitta\_no' is found in the splitted data:

**Assign** the second element of the splitted data to 'name\_of\_city'

**Assign** the third element of the splitted data to 'direction'

**Assign** the fourth element of the splitted data to 'area'

**Generate** a receipt using the ‘return\_invoice' function with parameters:

**Call** the 'write.rent\_land' function with 'kitta\_no' as a parameter

**Open** the file with the name '<name>.txt' in read mode and assign the value

**Print** the content of the file

**Exit** the loop

# Flowchart

Flowcharts are the chats that represents a process or a computer algorithm. Flowcharts consists of symbols that represents flow, start process of the system. In flow chart the parallelogram represents the statements and rectangle represent a process or a calculation part of a program where as the lines are for the flow direction and the diamond shapes are for the control flow of the process. Flowcharts are often called as process map. It is in use for both of the technical as well as non-techincal people in numerous fields. (chart, 2023)

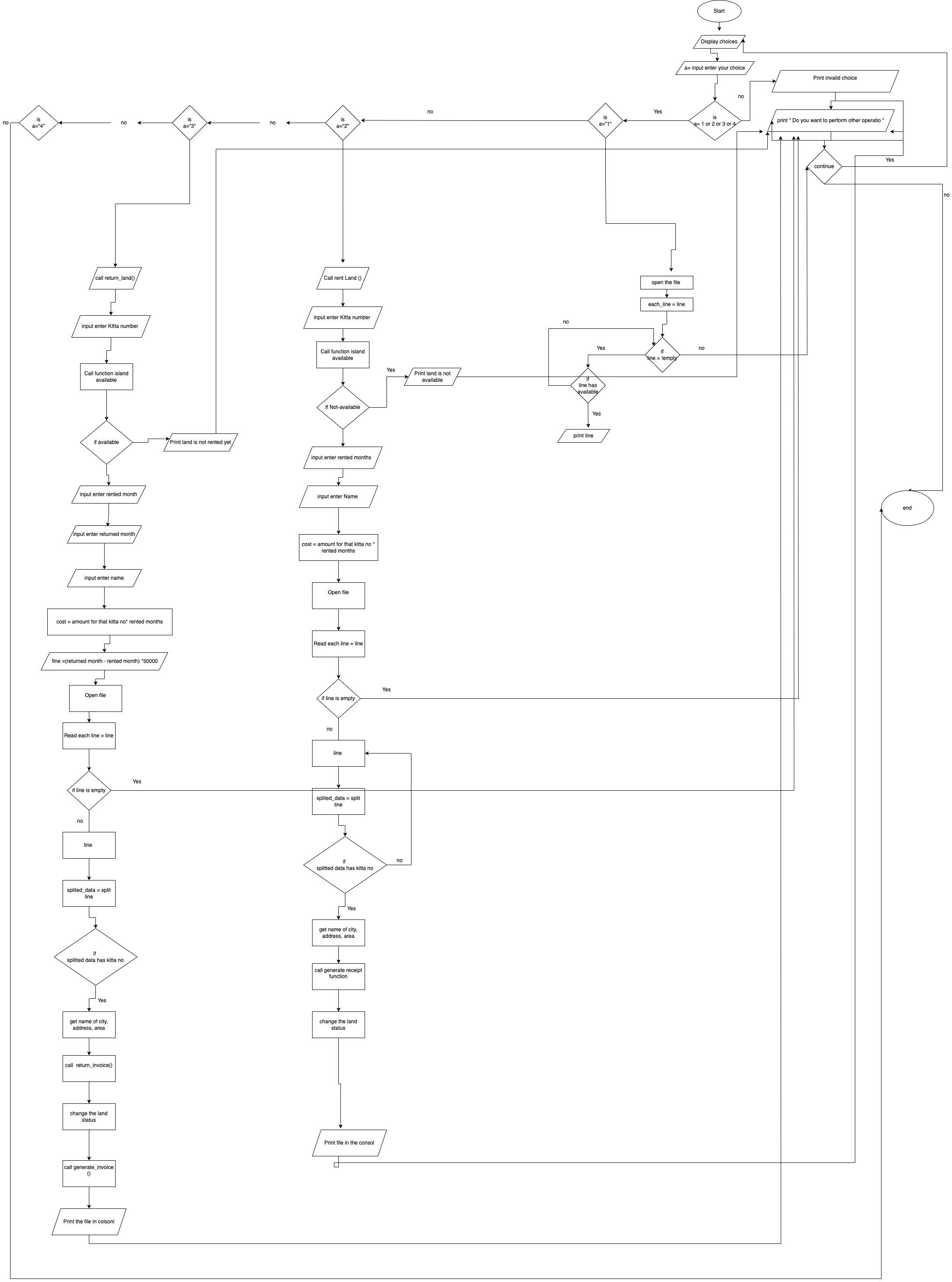


Figure 5: flowchart of the program

# Data Structure

Data structure is the way of arranging or structuring the data such that it can be accessed easily and efficiently at the time of need. Data structures serve as the foundational building blocks for constructing programs. Data Structure supports operations like insertation, removal, shorting,searching of the data. There are mainly two types of data structures they are primitive data structure and non-primitive data structures.

**Primitive data**

**String:**

Strings are considerd as primitive data in python. It is a sequence of characters enclosed between two (‘) or two (“). This data can only be created once and cannot be edit once created as they are immutable.



Figure 6: String example screenshot

**Integer** **:**

Integers are primitive data type that holds numerical values. In pythone integer can can be as long as posssible. It uses key word int to be declared. Below is the example of integer data type used.



Figure 7: Integer example screenshot

**Boolean:**

boolean is the other [primitive data type in python. It give one of the two values that is either (True or false). In the coursework also I have used different boolean types but through different conditions.

**Non primitive data**

**List:**

List are the most flexible data structures in Python,which are simply a list of square bracket-enclosed which componetents are seperated by comma. It is possible for a list to include both homogeneous and heterogeneous components such as it can hold data of same type as well as data’s which are of different types. Index(), append(), extend(), insert(), delete(), pop(), and other methods are available on a List. Lists can have their content changed without losing their identity since they are modifiable.



Figure 8: List example screen shot

Dictonary:

Dictonaries are the other in built non primitive data structure supported by pythone. It consists of datas with many keys and many keys can hold large amount of datas in it. The data’s in dictonary are ordered as well as mutable.

A close-up of a white background

Description automatically generated

Here, In the course work I have used python’s inbuilt data structure “List” for data storing and retrival. Python lists are an ordered collection of data, just like arrays, which are stated in other languages. In the list any type of dissimmilar data can be stored and retrived. The key feature of list are:

1. Lists are mutable we can append, or remove the data in it even after the creation of it is comleted.
2. List also allows duplication which gives user chance to have multiple iteams with same value

Here in the courswork I have used list to store data from the file and after manupulation of the data again list is used to append the data in the file.

# Program

This python program is designed to ease the rental services of the lands. There are 4 options a user can choose in the system that are view land, rent land , return land and exit. By selecting the corresponding number a user can select the need. For offering the options the main() functions is called and the options are printed for the user. When the user chooses 1 available lands are displayed and if user chooses the option 2 or 3 the user input is printed for the rental or return of the land. If the user chooses 4 as a option the program is end but if the user chooses other inputs suitable message is displayed

* Main:

This where the program basically starts using this function the user is asked to enter the choice and the if -else as well as try except checks the user input and calls the necessary function for the action to be taken.

* Available land:

If the user selects opton 1 the available\_land() function comes into action Using this function all the availble lands are shown to the user along with its respective informations like kitta no , direction, area, etc are shown to user

* Rent function

When the user chooses the option 2 this function{rent\_land()} is called. The rent\_land function assists with the renting process of the land by accepting inputs like kitta number, rented months, name of the customer. This function also checks for the availability of land by calling is\_land\_rented() function. Here the cost is also calculated using the corresponding amount form the kitta no in a file with the numbers of rented months multiplyed using the cost() function. This function also calls generate receipt function which opens a file with cutomer name and appends the information relating to the purchase as well as it changes the statues from avialable to not – available using the funtion rent\_land from write.py file.

* Return function

When the user goes for the third option that is return land the return\_land() function is executed. The return\_land function assists with the return process of the land by accepting inputs like kitta number, rented months, returned months, name of the customer etc it also checks for the the land status if its already available it shows message to the user that the land is not rented yet. This function also calls the claculate\_cost() function to calculate cost as well as calculate\_fine function to calculate\_fine(). It then open a file named same as the customer name in append mode and writes the information relating to the return of the land



Figure 9: rent of the land screen shot

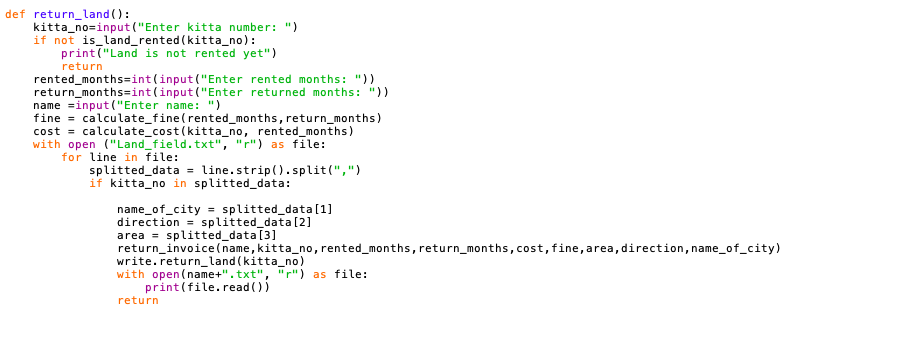


Figure 10: return of the land

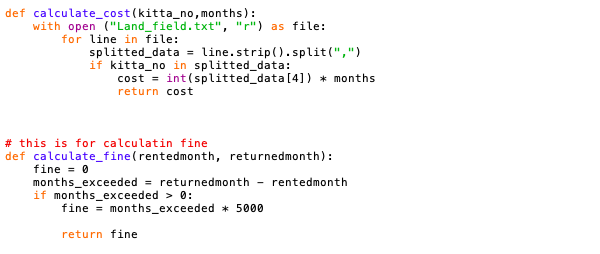


Figure 11: fine calculation and cost calculation

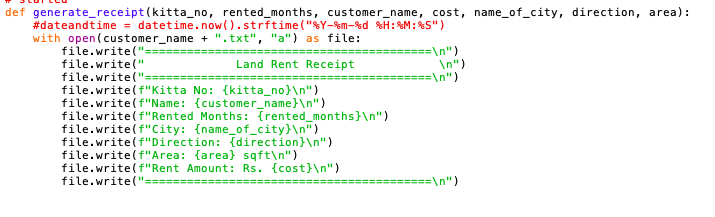


Figure 12: generation of text file for renting land

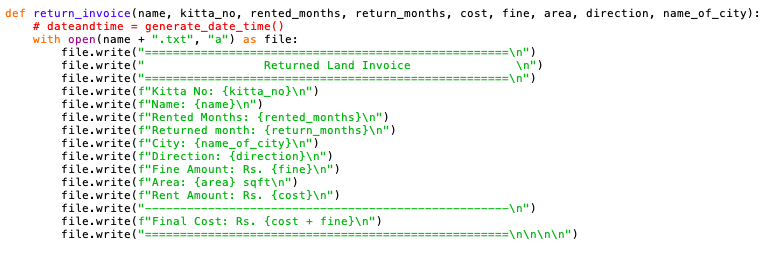


Figure 13: generating text file for the return of land

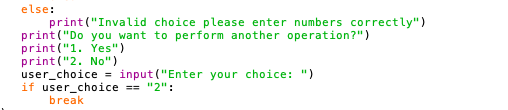


Figure 14: termination of program after selecting option

# Testing

## Test 1 Show implementation of try, except

Table 1: test 1 showing implementation of try except

|  |  |
| --- | --- |
| Objective | To Provide invalid inputs and display the message |
| Action | Choice is entered in string which is incorrect |
| Expected result | Suitable message must be displayed in terminal |
| Actual result | Suitable message is displayed in the terminal |
| conclusion | Test was successful |

Evidence:

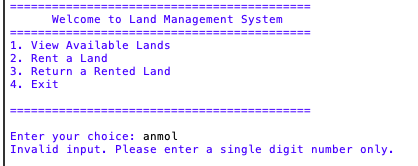


Figure 15: evidence for the implimentation of try , except

## Test 2 Selection rent and return of lands

Table 2 :Test 2 Selection rent and return of lands

|  |  |
| --- | --- |
| Objective | Selection rent and return of lands |
| Action | Provided negative value as input  Provided non existed value as input |
| Expected result | Invalid message must be displayed in the terminal |
| Actual result | Invalid message was displayed in the terminal |
| conclusion | Test was successful |

**Evidence:**

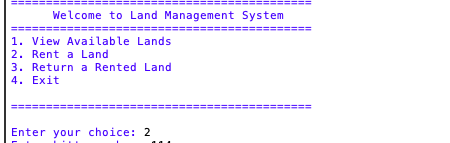
****

Figure 16: choosing to select rent



Figure 17:negative input Kitta no



Figure 18: output after entering negative kitta\_no number

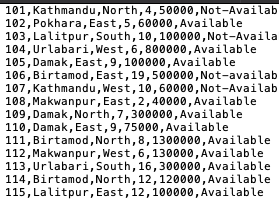


Figure 19: existing kitta\_number's



Figure 20: entering non existing value of kitta\_no



Figure 21: Message after entering non existing value

## Test 3 File generation of renting of land(s) (Renting multiple land(s))

Table 3: test-3 File generation of renting of land(s) (Renting multiple land(s))

|  |  |
| --- | --- |
| Objective | To rent and return multiple lands and genrate .tx |
| Action | Rent two lands simultaniously with same customer |
| Expected result | Both rented lands must be displayed in .txt file |
| Actual result | Both of the rented lands were displayed in the .txt file |

Evidence:

A screen shot of a computer

Description automatically generated

Figure 22: choosing option to rent land



Figure 23: inserting inputs for renting kitta\_no 110

A close-up of a receipt

Description automatically generated

Figure 24: shell output for renting the first land

A screenshot of a computer

Description automatically generated

Figure 25:Output for the fist land rented in text file

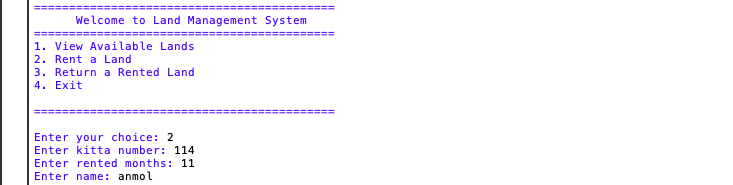


Figure 26: inputs for second Land with kitta\_no 114

A screenshot of a computer screen

Description automatically generated

Figure 27: output of multiple land rented in shell

A screenshot of a computer

Description automatically generated

Figure 28 Output for multiple renting in .txt file

## Test 4 File generation of returning process of land(s) (Returning multiple land(s))

Table 4: test-4 File generation of returning process of land(s) (Returning multiple land(s))

|  |  |
| --- | --- |
| Objective | File generation of returning process of land(s) (Returning multiple land(s)) |
| Actions | Returned multiple lands |
| Expected result | Multiple returned lands displayed in text file |
| Actual result | Multiple returned lands were displayed in text file |
| Coclusion | The test was successful |

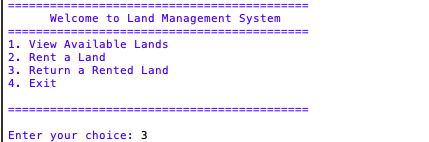
Evidance:

Figure 29: choosing returning option



Figure 30: entering inputs for returning land with kitta no 103

A close-up of a computer screen

Description automatically generated

Figure 31: shell output as invoice for returning land 103(first return)

A screenshot of a computer

Description automatically generated

Figure 32: output for the first land returned by same customer



Figure 33: inputs for returning second land of kitta\_no 107 by same customer

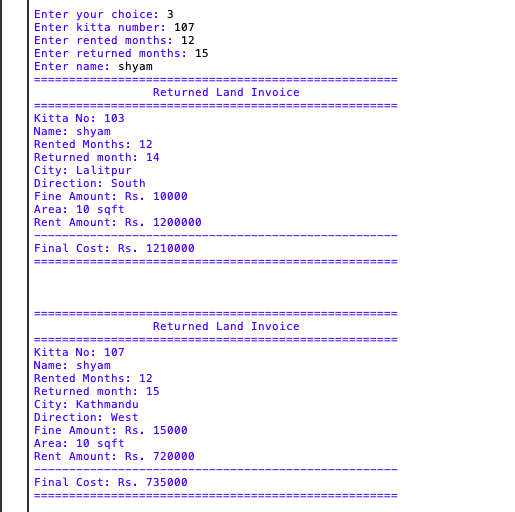


Figure 34: invoice printed in shell for returning 2 lands by same person

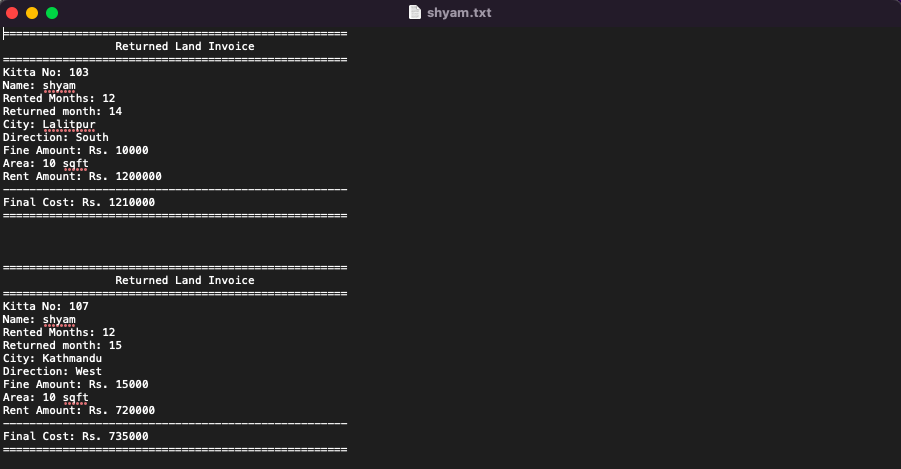


Figure 35: output invoice of returning multiple land in .txt file

Test 5 Show the update in stock of land(s)

Table 5 Test-5 Show the update in stock of land(s)

|  |  |
| --- | --- |
| Objective | To Show the update in stock of land(s) |
| Actions | Rented a aviailable land  Returned a not available land |
| Expected result | The status in the land data textfield must be changed to Available from Not-Available during the returning and change Available to Not- Available during the renting |
| Actual result | The status was successfully changed from available to Not-Available when renting and Not-Available to available while returning |
| Coclusion | The test was successful |

Evidence:

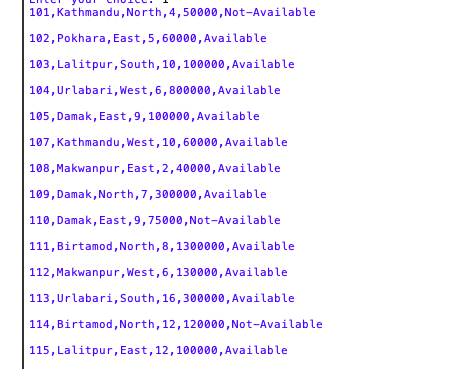


Figure 36: Land data info in shell

A screenshot of a computer

Description automatically generated

Figure 37: land field data in .txt file



Figure 38: choosing to rent this(Before renting)

A white background with blue text

Description automatically generated

Figure 39: choosing rent option

A close-up of a receipt

Description automatically generated

Figure 40: successful rent

A screenshot of a computer program

Description automatically generated



Figure 41 after rent was successful status changed in shell

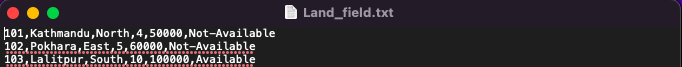




Figure 42: status changed in .txt file after successful rent



Figure 43: before returningof land in shell

A black background with white text

Description automatically generated

Figure 44: before returning of land in .txt file

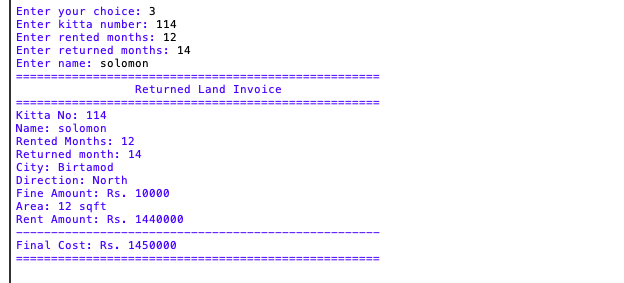


Figure 45: successful renturning of kitta no 114

A close-up of several blue text

Description automatically generated



Figure 46: change in status of land in shell after successful returning

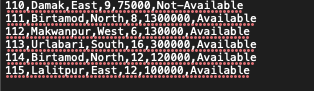




Figure 47: changing status of land in file after successful returning

# Conclusion

Finally, the python course has provided us with the knowledge of python programming language, its wide variety of application and surface understanding of OOP . Through out the course we covered wide varieties of topics from data types function to file handling and from control flow to class and function. During the course we had also gained sound knowledge on problem solving and algorithms as we had applied these concept to real world challenges. This course had also provided us with concept of modular programming and concepts of oop. Overall, this course had provided us with solid foundation for beginning programmer like me to have in-depth research for the development of the other complex programs.

As I only had the basic concepts of java I found it difficult to adjust with the program flow during the start of my work. The other problem I faced during my work was unorganized planning to find the start the work. Although I faced difficulty at first I overcame all those from research on internet I even wached some youtube clips for python tips. I am grateful towards our module leader and teachers who had provided me with necessary guideline to complete my work

Overall, I enjoyed python programming and had a chance to add another language to my portfolio.

# References

(Alexander S. Gillis, n.d.)

(Foundation, 2024)

(UpGuard, 2024)

(chart, 2023)

(Coursera, apr 4 2024)

# Appendix

def main(): #this is the main function for the program and also the entry point

while True:

print("\n===========================================")

print(" Welcome to Land Management System")

print("===========================================")

print("1. View Available Lands")

print("2. Rent a Land")

print("3. Return a Rented Land")

print("4. Exit\n")

print("===========================================\n")

try:

user\_choice = input("Enter your choice: ")

if user\_choice == "":

raise ValueError

user\_choice = int(user\_choice)

if user\_choice < 0 or user\_choice > 9:

raise ValueError

except ValueError:

print("Invalid input. Please enter a single digit positive number only.")

continue

if (user\_choice == 1):

available\_land('Land\_field.txt')

elif (user\_choice == 2):

rent\_land()

elif (user\_choice == 3):

return\_land()

elif (user\_choice)== 4:

break

else:

print("Invalid choice please enter numbers correctly")

print("Do you want to perform another operation?")

print("1. Yes")

print("2. No")

user\_choice = input("Enter your choice: ")

if user\_choice == "2":

break

main()

import write

# Function to check if a land is available for rent

def is\_land\_available(kitta\_no):

with open("Land\_field.txt", "r") as file:

for line in file:

splitted\_data = line.rstrip().split(",")

if kitta\_no in splitted\_data and splitted\_data[-1] == "Available":

return True

return False

# Function to check if a land is already rented

def is\_land\_rented(kitta\_no):

with open("Land\_field.txt", "r") as file:

for line in file:

splitted\_data = line.rstrip().split(",")

if kitta\_no in splitted\_data and splitted\_data[-1] == "Not-Available":

return True

return False

# Function to calculate the cost of renting a land for a given number of months

def calculate\_cost(kitta\_no,months):

with open ("Land\_field.txt", "r") as file:

for line in file:

splitted\_data = line.strip().split(",")

if kitta\_no in splitted\_data:

cost = int(splitted\_data[4]) \* months

return cost

# this is for calculatin fine

def calculate\_fine(rentedmonth, returnedmonth):

fine = 0

months\_exceeded = returnedmonth - rentedmonth

if months\_exceeded > 0:

fine = months\_exceeded \* 5000

return fine

# Function to generate a rent receipt for a rented land

def generate\_receipt(kitta\_no, rented\_months, customer\_name, cost, name\_of\_city, direction, area):

with open(customer\_name + ".txt", "a") as file:

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

file.write(" Land Rent Receipt \n")

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

file.write(f"Kitta No: {kitta\_no}\n")

file.write(f"Name: {customer\_name}\n")

file.write(f"Rented Months: {rented\_months}\n")

file.write(f"City: {name\_of\_city}\n")

file.write(f"Direction: {direction}\n")

file.write(f"Area: {area} sqft\n")

file.write(f"Rent Amount: Rs. {cost}\n")

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

# Function to check if a kitta number exists in the Land\_field.txt file

def kitta\_exists(kitta\_no):

with open("Land\_field.txt", "r") as file:

for line in file:

splitted\_data = line.rstrip().split(",")

if kitta\_no in splitted\_data:

return True

return False

# Function to handle the return of a rented land

def return\_land():

kitta\_no=input("Enter kitta number: ")

if not kitta\_exists(kitta\_no):

print("Kitta number does not exist")

return

if(int(kitta\_no)<0):

print("Kitta number should be positive")

return

if not is\_land\_rented(kitta\_no):

print("Land is not rented yet")

return

rented\_months=int(input("Enter rented months: "))

return\_months=int(input("Enter returned months: "))

name = str(input("Enter name: "))

fine = calculate\_fine(rented\_months,return\_months)

cost = calculate\_cost(kitta\_no, rented\_months)

with open ("Land\_field.txt", "r") as file:

for line in file:

splitted\_data = line.strip().split(",")

if kitta\_no in splitted\_data:

name\_of\_city = splitted\_data[1]

direction = splitted\_data[2]

area = splitted\_data[3]

return\_invoice(name,kitta\_no,rented\_months,return\_months,cost,fine,area,direction,name\_of\_city)

write.return\_land(kitta\_no)

with open(name+".txt", "r") as file:

print(file.read())

return

# Function to generate an invoice for a returned land

def return\_invoice(name, kitta\_no, rented\_months, return\_months, cost, fine, area, direction, name\_of\_city):

with open(name + ".txt", "a") as file:

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

file.write(" Returned Land Invoice \n")

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

file.write(f"Kitta No: {kitta\_no}\n")

file.write(f"Name: {name}\n")

file.write(f"Rented Months: {rented\_months}\n")

file.write(f"Returned month: {return\_months}\n")

file.write(f"City: {name\_of\_city}\n")

file.write(f"Direction: {direction}\n")

file.write(f"Fine Amount: Rs. {fine}\n")

file.write(f"Area: {area} sqft\n")

file.write(f"Rent Amount: Rs. {cost}\n")

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

file.write(f"Final Cost: Rs. {cost + fine}\n")

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

# Function to handle the renting of a land

def rent\_land():

kitta\_no=input("Enter kitta number: ")

if not kitta\_exists(kitta\_no):

print("Kitta number does not exist")

return

if(int(kitta\_no)<0):

print("Kitta number should be positive")

return

if is\_land\_rented(kitta\_no):

print("Land is not available")

return

rente\_months=int(input("Enter rented months: "))

name =input("Enter name: ")

cost = calculate\_cost(kitta\_no, rente\_months)

with open ("Land\_field.txt", "r") as file:

for line in file:

splitted\_data = line.strip().split(",")

if kitta\_no in splitted\_data:

name\_of\_city = splitted\_data[1]

direction = splitted\_data[2]

area = splitted\_data[3]

generate\_receipt(kitta\_no,rente\_months,name,cost,name\_of\_city,direction,area)

write.rent\_land(kitta\_no)

with open(name+".txt", "r") as file:

print(file.read())

return

def read\_file(file\_name):

with open('file\_name.txt','r') as file:

anmol = file.readline()

print(anmol)

def rented\_lands(file\_name):

with open('file\_name.txt', 'r') as file:

for line in file:

splitted\_data = line.strip().split(",")

if "Not-available" in splitted\_data:

print(line)

def available\_land(file\_name):

with open(file\_name,'r') as file:

each\_line=file.readlines()

for line in each\_line:

if "Available" in line:

print(line)

def rent\_land(kitta\_no):

with open("Land\_field.txt", "r+") as file:

lines = file.readlines()

for index, line in enumerate(lines):

splitted\_data = line.strip().split(",")

if kitta\_no in splitted\_data:

if splitted\_data[-1]=="Available":

splitted\_data[-1]="Not-Available"

lines[index] = ",".join(splitted\_data) + "\n"

file.seek(0)

file.writelines(lines)

file.truncate()

else:

print("Land is already rented")

def return\_land(kitta\_no):

with open("Land\_field.txt", "r+") as file:

lines = file.readlines()

for index, line in enumerate(lines):

splitted\_data = line.rstrip().split(",")

if kitta\_no in splitted\_data:

if splitted\_data[-1]=="Not-Available":

splitted\_data[-1]="Available"

lines[index] = ",".join(splitted\_data) + "\n"

file.seek(0)

file.writelines(lines)

file.truncate()

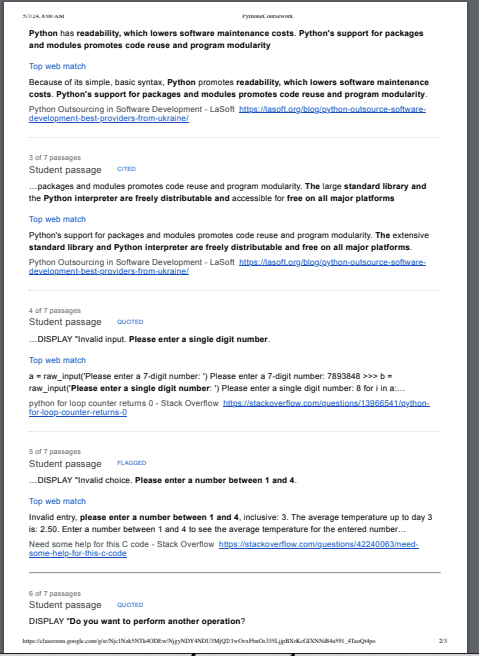
else:

print("Land is already available")

# Plagiarism report

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated