A logo for a university

Description automatically generated



**CS4051NI Fundamentals of Computing**

**60% Individual Coursework**

**2023/24 Spring**

**Student Name:** Arpit Neupane

**London Met ID:** 23050197

**College ID:** NP01CP4A230331

**Assignment Due Date:** 7 May, 2024

**Assignment Submission Date:** 6 May, 2024

**Word Count: 242**

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

# Table of Content

# List of Figures

# List of Tables

# INTRODUCTION

Welcome to the documentation of the Python File-Handeling project, designed specially for managing land rental related transactions. This documentation/report is a guide to understanding the functionalities and features of the project, providing a sneak peek; no matter how simple it may be; into its design and implementation.

Data refers to basic facts and figures. It is generally raw information that simply includes basic numbers or texts. It’s the fundamental building block of any system, comprising of raw facts and figures. Information is known to be the processed form of data. Data/Information management is the process of organizing, storaging, and manipulating that information to make sure of its accuracy, accessibility, as well as it’s security. (techtarget)

File handeling is a way to better manage your data. File handling allows one to create different types of files of many formats that can store and manipulate data for your programs such as .txt, CVS files, JSON files and so on. Text files store data as plain text, while CSV and JSON are widely-used formats for storing structured data. There are 6 modes in which files can be opened. They are: read, write, append, read+, write+ and append+. (https://datatrained.com/post/file-handling-in-python/#:~:text=File%20handling%20in%20Python%20allows%20us%20to%20store%20data%20that,be%20performed%20on%20the%20file)

This project oversees land data and uses file handeling to effectively sell land to customers of the company TechnoPropertyNepal.

## Goal

The goal of this project is develop a basic, easy to use, functional application that efficiently manages land rental transactions, effectively simplifying the process of renting land for both the company and tenants. Through the effective use of technology, the project aims to enhance the overall experience of renting land, making it more accessible and convenient.

## Objectives

1. **Simplifying Land Rental Processes**: The foremost objective of this project is to work towards simpifying the land rental as well as land returning process, while also eliminating any complicated steps and making it easier for people to rent land ; effectively saving time and effort.
2. **Maintaining Accurate Transaction Records**: Another significant objective of the project is to ensure that both the tenants and the company has access to accurate information. Reliability of company is also the focus of the project.
3. **User- friendly Design**: While building the project, the user experience and user interface was taken deeply into account. The UI is simple and easy to use and hence the user is not overwhelmed by the available information.
4. **Promoting Transparency**: Transparency and clarity should always be one of the most focused part when developing any interface, but is often overlooked. At every step, the user is provided with updated information, and hence clarity and transparency is promoted.

# Discussion and Analysis

## Selected Text Editor

The text editor used while developing this simple land rental application was the default IDE(Integrated Development Environment) that comes with python, IDLE(Integrated Development and Learning Environment).

Reference this: https://docs.python.org/3/library/idle.html

IDLE has the following features that makes it download-worthy:

* It’s cross-platform which means it works mostly the same on Windows, Unix, and macOS
* Python interactive interpreter with colorizing of code input, output, and error messages, which makes it easier to code
* Text editor with multiple features like multiple undo, python colorizing, smart indent, call tips, auto completion, and much more.
* search within any window and multiple files (grep)
* Has a built-in debugger with persistent breakpoints, stepping, and viewing of global and local namespaces



## Algorithm

Basically, an algorithm is a step by step process to solve a problem. While algorithms may not be required for smaller programs, they are one of the most reliant tools when building large-scale programs aand/or solving large-scale problems.

The algorithm of program is as follows:

**Step 1**: Start the Land Rental System.

**Step 2**: Read data from “land.txt” file and store in dictionary landDict.

**Step 3**: Print the land details from landDict in tabular form.

**Step 4**: Print the options that is available to the user (Rent, Return and Exit).

**Step 5**: Take input from the user and store it in userChoice.

**Step 6**: If userChoice is 1, go to **Step 7**, elif userChoice is 2, go to **Step 15**, elif userChoice is 3, go to **Step 23**.

**Step 7**: Prompt user to input information such as name, address, contact info, kitta number to rent, anna s/he wants to rent, number of months to rent and store it in variables userName, address, phoneNumber, kittaId, anna, rentMonth respectively.

**Step 8**: Store data provided to dictionary dict\_ , key:kittaId, and values:other datas.

**Step 9**: Calculate the total amount by multiplying rentMonth and price.

**Step 10**: Prompt the user if he wants to rent again.

**Step 11**: If yes, go to **step 7**, else go to **step 12**.

**Step 12**: Print bill according to the given information of user and rented land.

**Step 13**: Write exact printed bill to unique-named file.

**Step 14**: Change availability of .txt file to “ Not Available”.

**Step 15**: Prompt user to input information such as name, address, contact info, kitta number to return, number of rented months, number of months in which land was returned, and store it in variables userName, address, phoneNumber, kittaId, rentMonth, returnMonth respectively.

**Step 16**: Store data provided to dictionary dict\_ , key:kittaId, and values:other datas.

**Step 17**: If returnMonth is greater than rentMonth, totalPrice is multiplcation of returnMonth and price and extra 10% fine for each month, else totalPrice is multiplcation of returnMonth and price.

**Step 18**: Prompt the user if he wants to return again.

**Step 19**: If yes, go to **step 15**, else go to **step 20**.

**Step 20**: Print bill according to the given information of user and returned land.

**Step 21**: Write exact printed bill to unique-named file.

**Step 22**: Change availability of .txt file to “Available”.

**Step 23**: Stop the Land Rental System.

## Flowchart

Flowchart is a diagrammatical or visual representation of an algorithm. It is easier to visualize the flow of a problem or a program in a flowchart than an algorithm. It also separates the different types of actions into different shapes which makes the visualization process much easier to grasp.

The flowchart of the program is as follows:

# Program

This section explains how the overall program works in details as well as the features that this program has to offer.

## Implementation

The assignment was about developing a management system for land renting related transactions. The program’s functionality was broken down into 4 modules, each serving a specific purpose.

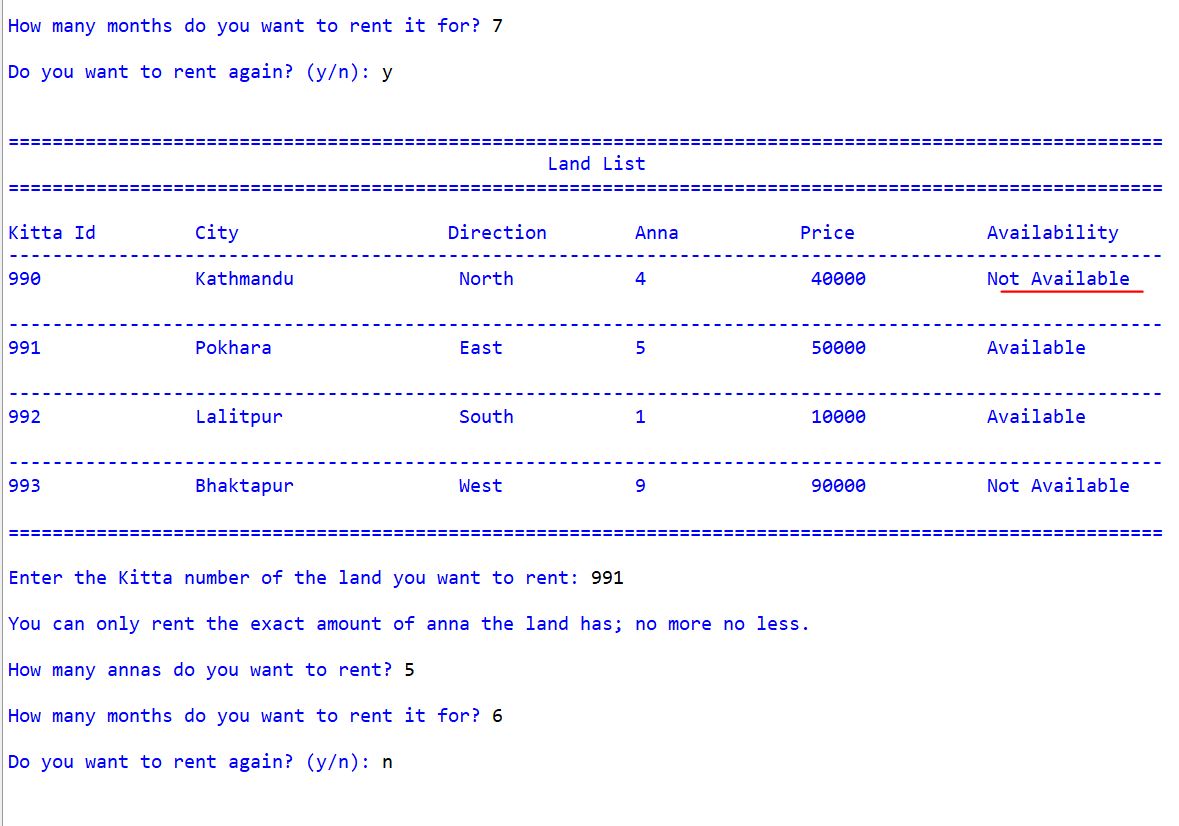
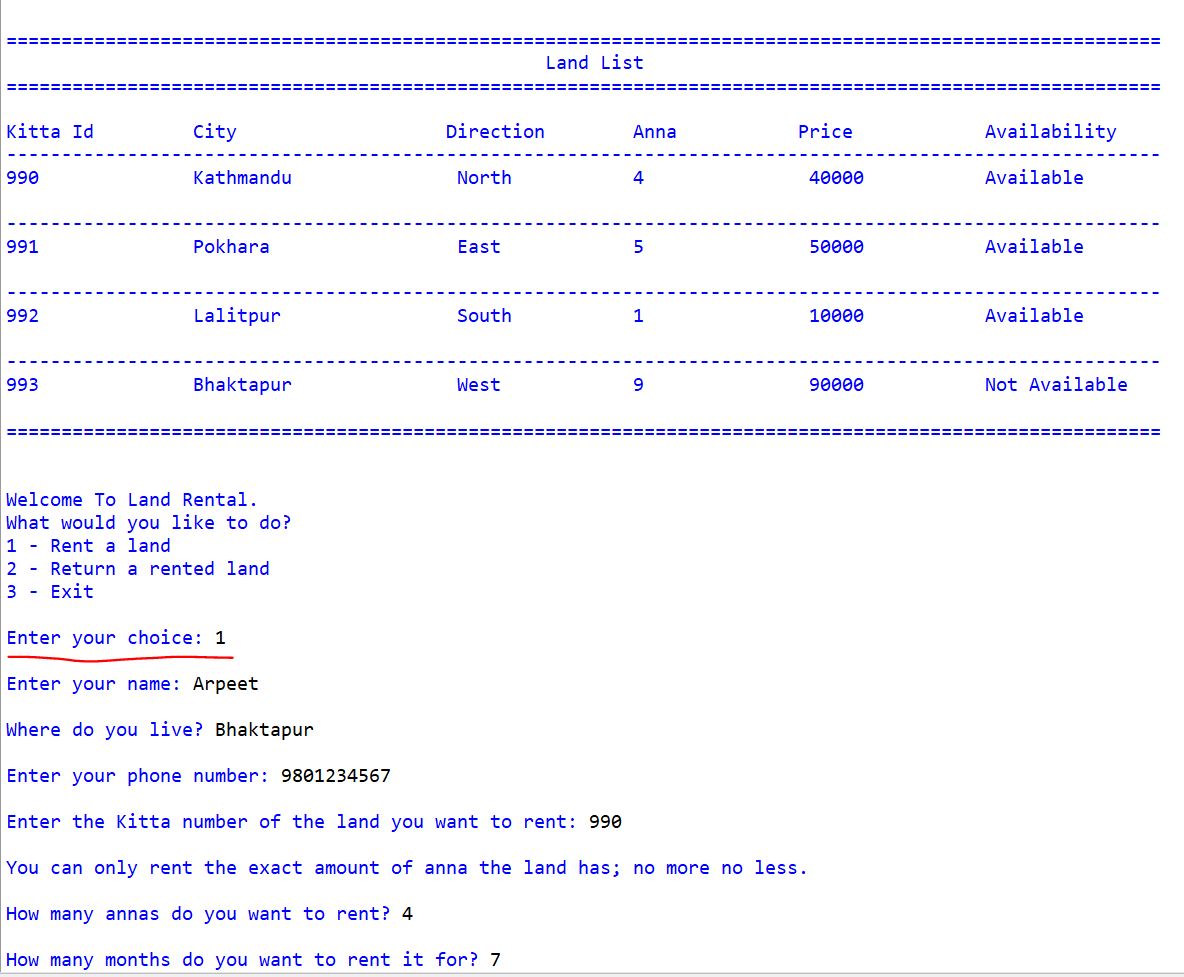
**read.py**

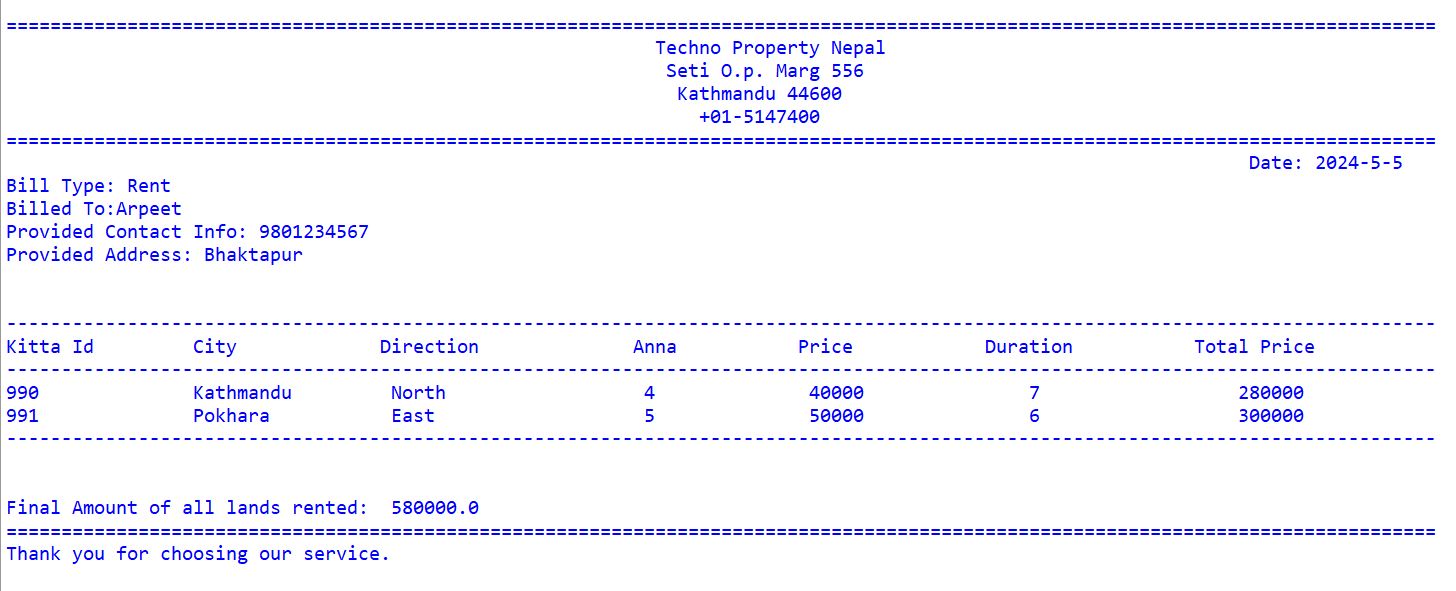
Like a foundation of a house, the first module, read.py is like a base of the program. It has a function called readingFile() which reads data from a text file called land.txt. An empty dictionary is initialized and after splitting the items of the datas in land.txt on the basis of comma(,), the 0th index value is put as key of the dictionary. An empty list is also initialized, and all the other datas are kept as datas of the list after removing the escape sequence(\n). Then, the corresponding key, value pairs are connected. The function has no parameter and it also returns the dictionary so that it can be used in other modules as required.

**operations.py**

This module is where most of the core logic of the program is implemented. It has 3 functions namely; renting, returning and fineForContractBreachers. Just as the name suggests, these functions are for specific purposes of renting, returning and applying fine to those who breach the contract. The code starts off with the logic of renting, where the user is promted to ask his/her name, address, contact information. A while loop is created before asking the id of the land they’d like to rent so that if user wants to rent again, the next time user is only asked for kitta id, anna and duration of rent. Only after making sure that the land is available, the anna(for confirmation)and the duration of rent is asked. All of these have been encapsulated(not in the programming sense of encapsulation, more like covered) in try except blocks in case the program ends due to an error and also so that the user experience is smooth.

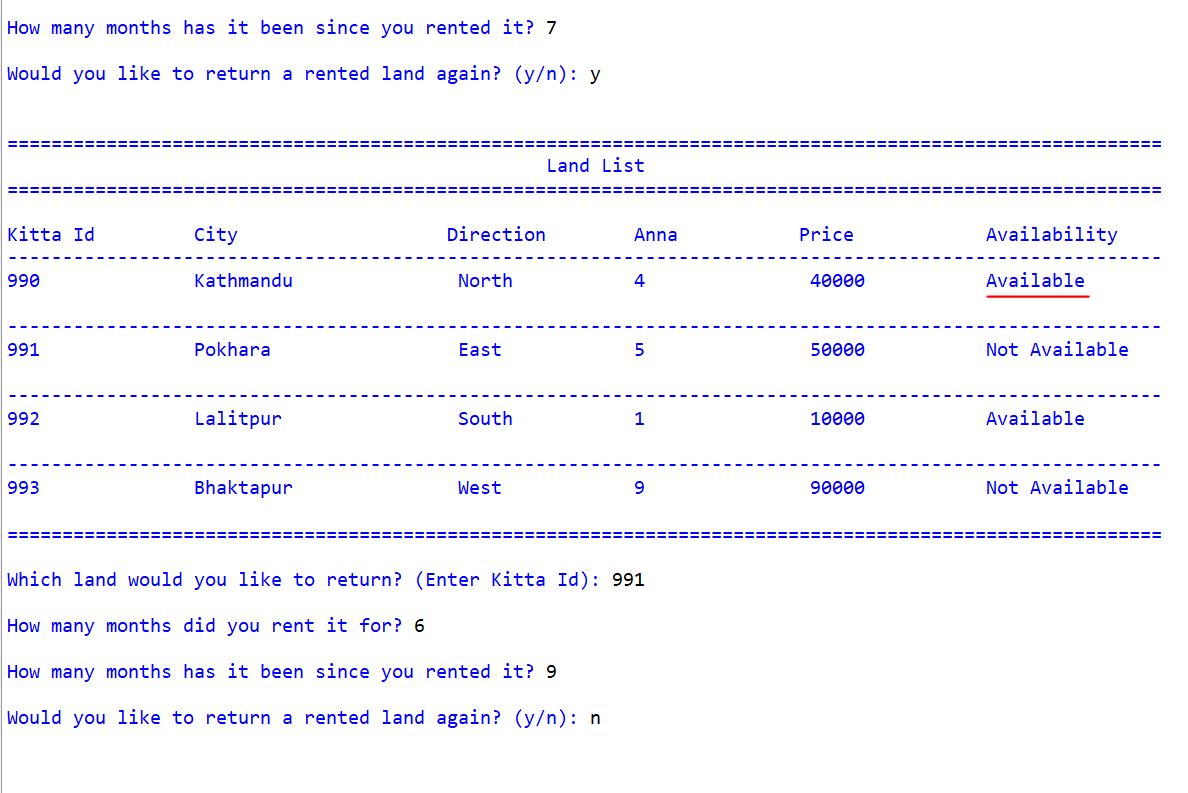
Then, in the dictionary’s (where the land datas have been stored) availablility section, the land which has been rented, is changed to “ Not Available”, so that when the next user wants to rent it, it won’t be available. The total price is calculated by multiplying the base price of the land with the duration of rent. Then, all of this data is put into another dictionary, where the user provided kitta id is the key and other datas are values. After that, the user asks if the user wants to rent again. If s/he chooses to do so, the land list is shown to the user(with changes to the availablity) and the above process repeats from asking kitta id. If not, the bill is printed and and availability is changed(to the text file). If the land is not available or user inputs land id which is not in the hands of the company, appropriate message is displayed which will further be talked about in the testing section.





The second function is for returning the rented land, which has pretty much the same concept as the renting function, asking the user for personal and rental information. Only one extra information is asked, which is the duration that the user has used the land for. This is used to calculate total amount as well as the fine which the 3rd function handles. If the user breaches the contract and returns land late, the total amount will be the multiple of number of months s/he has used the land and base price of land + a fine of 10%.

A screenshot of a computer

Description automatically generated

A close-up of a check

Description automatically generated

**write.py**

The write.py module has 4 functions which handles functionalities such as creating a unique file name based on current date and time as well as user name and type of bill(returned or rented), printing bills; one for renting land and one for returning land, creating a text file with the unique file name and writing the bill onto it. The remaining function changes the availability of the land to the text file after the land has been rented or returned.

As shown in the picture above, the rent-type bill firstly shows information about the company and then it has information such as the date the bill was generated, user’s name, phone number and address and finally in tabular form: Kitta id, City, Direction Faced, Anna, Base Price, Duration, Total Price and Grand Total. All this info is exactly written to file with unique name generation process explained above.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The return-type bill has the exact same layout but with 3 extra information being Duration(returned month), Price after Fine and Grand Total after Fine.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**main.py**

The main.py module is where all the functionalities are called and also the module which is ran to access the whole program. Here, the land list is printed in tabular form and the user is promted to choose one of the following: to rent, to return or to exit corresponding to numbers 1, 2 and 3 respectively. If 1 is chosen, the renting function is called as explained in the operation.py section. If 2 is chosen, the returning function is called as explained in the operation.py section.

If the user chooses 3, s/he is exited from the program by making the main while loop False. If the user enters any other number, s/he is asked to try again and given guide to choose ne of 3 numbers(1, 2, or 3) and if the user enters a string, except block’s ValueError catches it, flags it as an error and the user is asked to choose an appropriate response again.

A screenshot of a computer screen

Description automatically generated

# Testing

## Test 1: Implementation of Try-except

|  |  |
| --- | --- |
| Objective | To check the effects of try-except block. |
| Action | The program is ran and value of datatype other than required is  inputted. (String is inputtted when int is required.) |
| Expected Result | The except block should triggered and ValueError should be caught. Code inside except should run. |
| Actual Result | As expected, the except block catches the ValueError. The code inside except block is executed showing appropriate guide to user. |
| Conclusion | The test was successful. |

### Image Proving the Test Result

A close-up of a computer screen

Description automatically generated

## Test 2.1: Selection of Options; Renting, Returning or Exit – Providing Negative Value

|  |  |
| --- | --- |
| Objective | To check input range of Kitta Id. |
| Action | A negative value is provided as input. |
| Expected Result | The input shouldn’t be accepted and Kitta Id should be asked again. |
| Actual Result | The input wasn’t be accepted and Kitta Id was asked again. |
| Conclusion | The test was successful. |

### Image Proving the Test Result

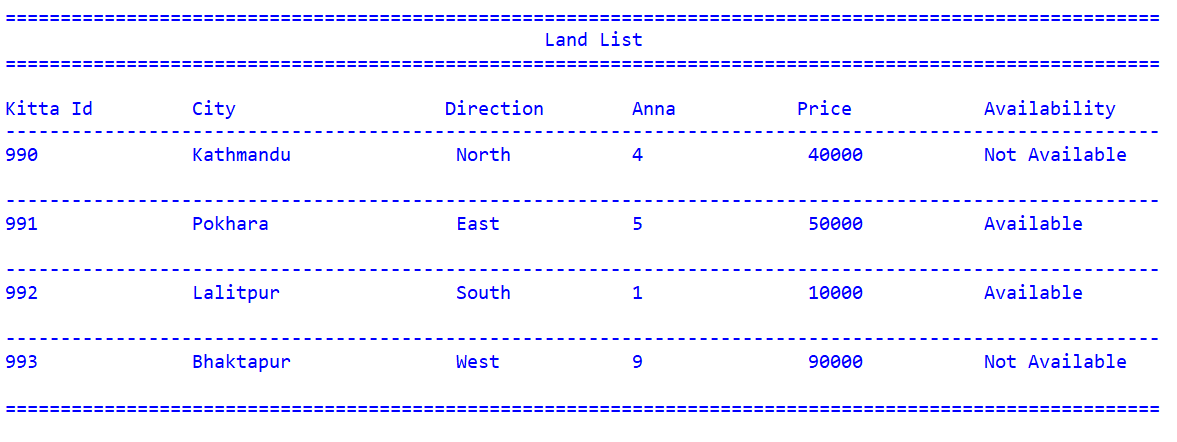
A screenshot of a computer

Description automatically generated

## Test 2.2: Selection of Options; Renting, Returning or Exit – Providing Non-existent Value

|  |  |
| --- | --- |
| Objective | To check input range of Kitta Id. |
| Action | A non-existent Kitta Id is provided as input. |
| Expected Result | The input shouldn’t be accepted and Kitta Id should be asked again. |
| Actual Result | The input shouldn’t be accepted and Kitta Id should be asked again. |
| Conclusion | The test was successful. |

### Image Proving the Test Result



A screenshot of a computer screen

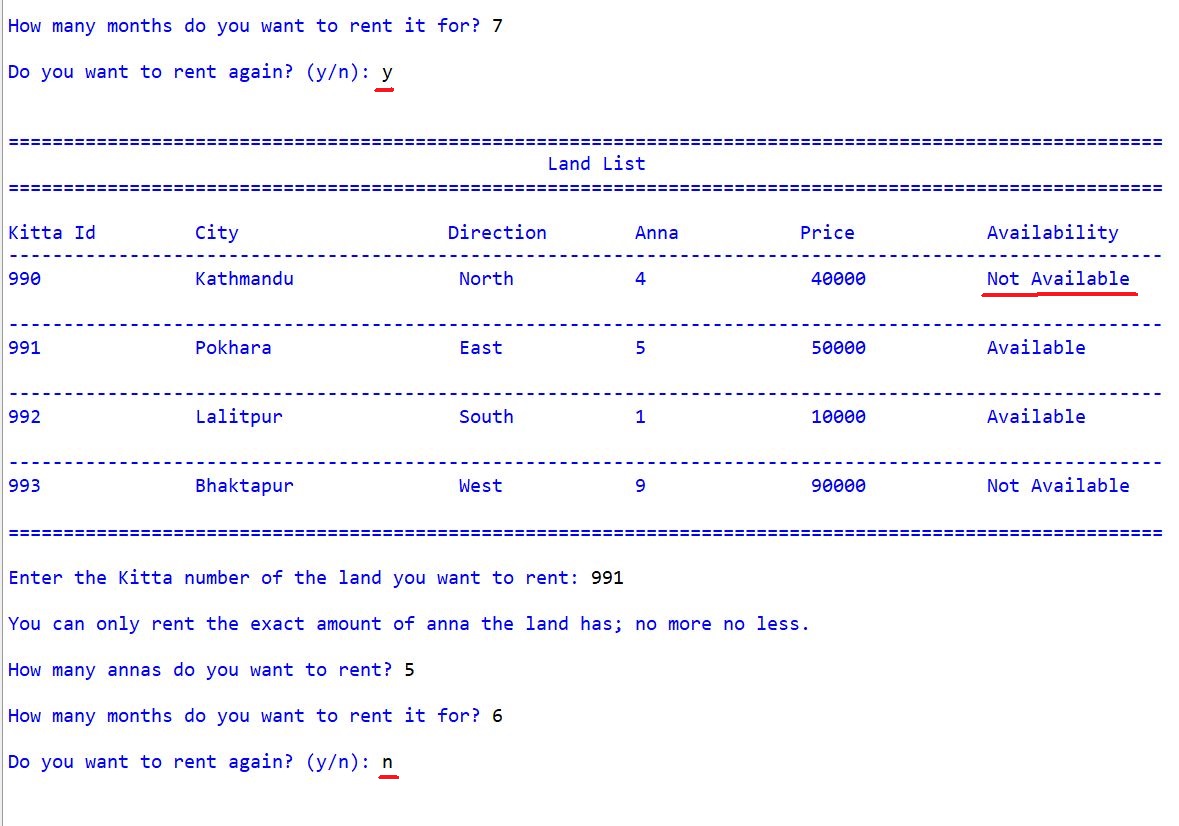
Description automatically generated

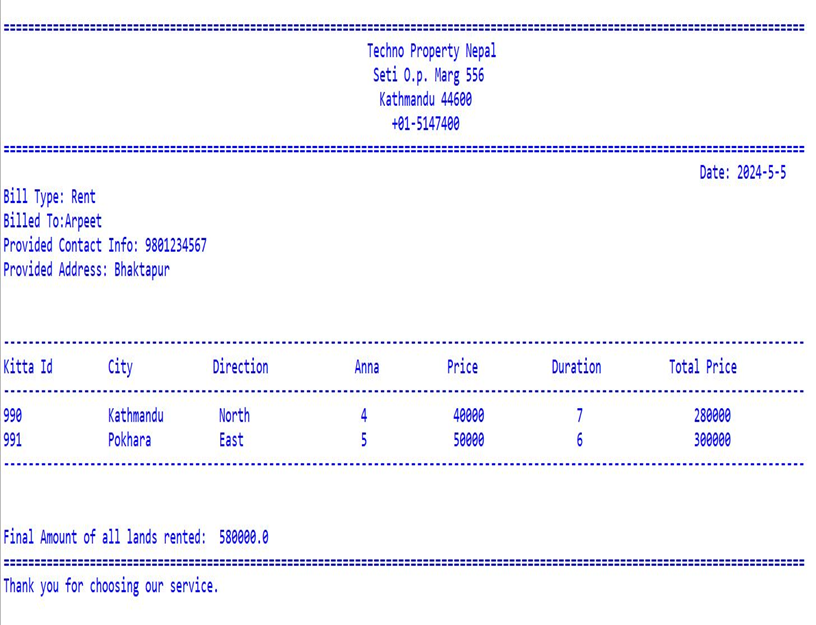
## Test 3: File Generation of Renting Multiple Lands

|  |  |
| --- | --- |
| Objective | To inspect the renting process and see the output in console as well as text file. |
| Action | Asked inputs such as name, kitta id, etc. is provided. |
| Expected Result | The bill should be printed in console and exact bill should be written in text file. |
| Actual Result | The bill was printed in console and exact bill was written in text file. |
| Conclusion | The test was successful. |

### Image Proving the Test Result

### A screenshot of a computer Description automatically generated





A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Test 4: File Generation of Returning Multiple Lands

|  |  |
| --- | --- |
| Objective | To inspect the returning process and see the output in console as well as text file. |
| Action | Asked inputs such as name, kitta id, etc. is provided. |
| Expected Result | The bill should be printed in console and exact bill should be written in text file. |
| Actual Result | The bill was printed in console and exact bill was written in text file. |
| Conclusion | The test was successful. |

### Image Proving the Test Result

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated



A close-up of a check

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

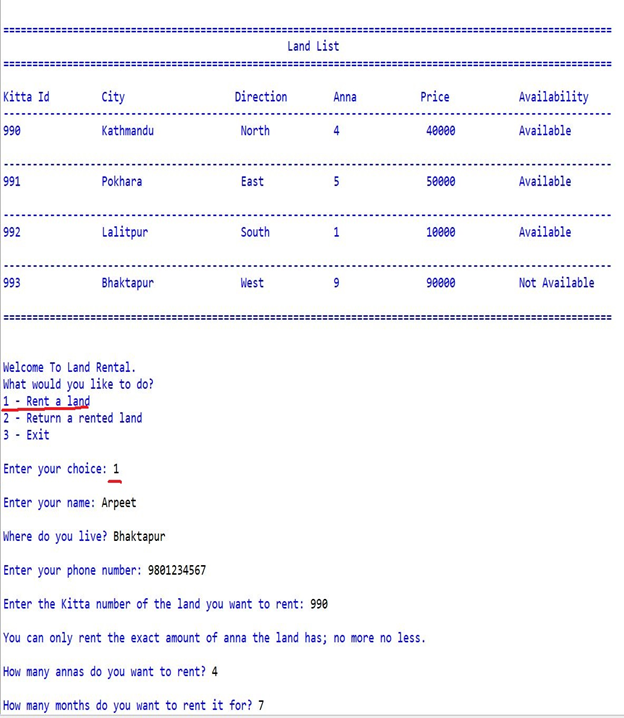
Description automatically generated

## Test 5.1: Changes to Availability of land after renting

|  |  |
| --- | --- |
| Objective | To inspect the changes that appear in the availability of the text file as well as the console after renting and returning land. |
| Action | Renting a land and then returning the same land. |
| Expected Result | After renting, in both the console and text file, the availability section should be updated to “ Not Available”.  After returning, in both the console and text file, the availability section should be updated to “ Available”. |
| Actual Result | After renting, in both the console and text file, the availability section was updated to “ Not Available”.  After returning, in both the console and text file, the availability section was updated to “ Available”. |
| Conclusion | The test was successful. |

### Image Proving the Test Result

**Renting**:





A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



**Returning**:

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



# Conclusion

Through learning, we attain new knowledge, skills, and values. It occurs sometimes naturally, through experiences; something akin to child learning to speak or a bird learning to fly. It can also be learned from someone we would consider a mentor or a teacher(imitational behaviour). (self-reference)

Through this coursework, I really learned a lot of ways to not only properly handle files in python, but many of the other concepts such as parameter and argument usage in functions, returning variables and booleans and using them in other functions, scopes of variable, which I had knowledge of but no practical experience; are now clear and engraved in my mind. Through the coursework, I’ve also expanded my research skills and through self-research and self-study, it feels as though I’ve learned a lot more than I would have through my teachers.

I would like to express my gratitude to the teachers and staff who came up with this coursework as well as my friends and colleagues who cleared me up on some concepts.

This coursework has equipped me with essential skills and ideas which I hope goes past just completing assignments and actually building enterprise level projects in the near future.

# Bibliography

# Appendix

## read.py

def readingFile():

"""Reads a file called land.txt and writes them in a list and a dictionary

There are no parameters in this function.

Returns:

Dictionary: landDict, keys are integers and values are lists.

"""

file = open("land.txt", "r")

lines = file.readlines()

landDict = {}

for line in lines:

line = line.replace("\n", "") #removing new line at the end of each line using replace function

list\_ = line.split(",") #splitting into parts wherever , is found

key = int(list\_[0]) #the first item of each line is assigned to variabe 'key'

value=[]

for i in range(1,len(list\_)):

value.append(list\_[i]) #list named 'value' has all datas except first one

landDict[key] = value #in dictionary landDict, each key is assigned it's value like it was in the text file

file.close()

return landDict

## operations.py

from read import readingFile

from write import rentBills

from write import returnBills

from write import changingAvailability

from write import changingAvailability

landDict = readingFile()

dict\_ = {}

def renting():

"""

This function asks users for information about them and the land they want to rent and the land is rented to the user.

"""

continueOperationBool = True

while continueOperationBool == True:

dict\_ = {} #making dict\_ empty for the next time the loop runs

#name - loop keeps running until user enters string

isAskedName = False

while isAskedName == False:

userName = input("\nEnter your name: ")

isString = True #assuming user inputted string

for char in userName:

if char not in "aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ":

isString = False

break

if isString == True and not userName == "" and not userName == " ":

#user cannot enter whitespace or null

isAskedName = True

else:

print("Please enter a valid name.")

#address - loop keeps running until user enters string

isAskedAddress = False

while isAskedAddress == False:

userAddress = input("\nWhere do you live? ")

isString = True #assuming user inputted string

for char in userAddress:

if char not in "aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ":

isString = False

break

if isString == True and not userAddress == "" and not userAddress == " ":

#user cannot enter whitespace or null

isAskedAddress = True

else:

print("Please enter a valid address.")

#making sure contact number is 10 digits long and non-string

isAskedContact = False

is10Numbered = False #assuming user didn't input 10 digits

while isAskedContact == False and is10Numbered == False:

contactNum = input("\nEnter your phone number: ")

isString = False

for chars in contactNum:

#converting each chars to lowercase to check ASCII value

if "a" <= chars.lower() and chars.lower() <= "z":

isString = True

break

if isString == True:

print("Enter non-string phone number.")

else:

#finally, only if non-string 10 digits are entered, the entered number is accepted

if len(contactNum) == 10:

is10Numbered = True

isAskedContact = True

else:

print("Enter 10 digits.")

#creating a loop that asks the user if s/he wants to rent again

rentAgainBool = True

while rentAgainBool == True:

#handeling ValueError of kittaId

try:

kittaId = int(input("\nEnter the Kitta number of the land you want to rent: "))

#checking if user inputted id is in list

if kittaId in landDict.keys() and landDict[kittaId][-1] == " Available":

#asking user for anna for validation

print("\nYou can only rent the exact amount of anna the land has; no more no less.")

isAskedAnna = False

while isAskedAnna == False:

try:

askingAnna = int(input("\nHow many annas do you want to rent? "))

availableAnna = int(landDict[kittaId][2])

if not askingAnna == availableAnna:

print("Only specified anna is available to rent.")

else:

isAskedAnna = True

except:

print("Value Error. Enter anna in numbers.")

#handeling ValueError of month

isAskedMonth = False

while isAskedMonth == False:

try:

monthZero = True

while monthZero == True:

rentsMonth = int(input("\nHow many months do you want to rent it for? "))

if rentsMonth <= 0:

print("Rent for at least one month.")

else:

monthZero = False

isAskedMonth = True

except ValueError:

print("ValueError. Enter month in numbers.")

#changing availability in dictionary so that the user can't rent same land again

landDict[kittaId][4] = " Not Available"

totalPrice = int(landDict[kittaId][3])\*rentsMonth

dict\_[kittaId] = [landDict[kittaId][0], landDict[kittaId][1], landDict[kittaId][2], landDict[kittaId][3], str(rentsMonth), str(totalPrice)]

repeatRent = True

while repeatRent == True:

rentAgain = input("\nDo you want to rent again? (y/n): ")

if rentAgain.lower() == "y":

#changing availability in text file so when list is printed, list is updated

changingAvailability(landDict)

#showing lands that the company controls in a list

print("\n")

print("="\*105)

print("\t\t\t\t\t\t Land List")

print("="\*105)

print("\nKitta Id \t City \t\tDirection Anna\t Price\t\t Availability")

file = open("land.txt", "r")

for line in file:

print("-"\*105)

#printing 105 \_

print(line.replace(",", "\t\t"))

#replacing , with double tab space and then printing file content

print("="\*105)

file.close()

rentAgainBool = True

repeatRent = False

#if user wants to rent again, this while loop wont run again, but upper loop to rent runs again

elif rentAgain.lower() == "n":

print("\n")

rentAgainBool = False

repeatRent = False

#if user doesn't want to rent again, this while loop and upper rent again loop wont run again

rentBills(dict\_, userName, contactNum, userAddress)

#calling function which changes data in land.txt from available to not available

#after user has rented

changingAvailability(landDict)

continueAgain = True

while continueAgain == True:

continueOperation = input("\nDo you want to continue to rent? (y/n): ")

if continueOperation.lower() == "y":

#showing lands that the company controls in a list

print("\n")

print("="\*105)

print("\t\t\t\t\t\t Land List")

print("="\*105)

print("\nKitta Id \t City \t\tDirection Anna\t Price\t\t Availability")

file = open("land.txt", "r")

for line in file:

print("-"\*105)

#printing 105 \_

print(line.replace(",", "\t\t"))

#replacing , with double tab space and then printing file content

print("="\*105)

file.close()

continueOperationBool = True

continueAgain = False

elif continueOperation.lower() == "n":

continueOperationBool = False

continueAgain = False

else:

print("Type option 'y' for yes and 'n' for no")

else:

print("Type option 'y' for yes and 'n' for no")

else:

if kittaId not in landDict.keys():

print("Our company doesn't have access to that land.")

else:

print("Sorry, that land has already been rented. Please rent a land that is available.")

except ValueError:

print("ValueError. Enter a valid Kitta number from the above list.")

def returning():

"""

This function asks users for information about them and the land they have rented and the land is returned to the company.

"""

continueOperationBool = True

while continueOperationBool == True:

dict\_ = {} #making dict\_ empty for the next time the loop runs

#name - loop keeps running until user enters string

isAskedName = False

while isAskedName == False:

returnUserName = input("\nEnter your name: ")

isString = True #assuming user inputted string

for char in returnUserName:

if char not in "aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ":

isString = False

break

if isString == True and not returnUserName == "" and not returnUserName == " ":

#user cannot enter whitespace or null

isAskedName = True

else:

print("Enter a valid name")

#address - loop keeps running until user enters string

isAskedAddress = False

while isAskedAddress == False:

userAddress = input("\nWhere do you live? ")

isString = True #assuming user inputted string

for char in userAddress:

if char not in "aAbBcCdDeEfFgGhHiIjJkKlLmMnNoOpPqQrRsStTuUvVwWxXyYzZ":

isString = False

break

if isString == True and not userAddress == "" and not userAddress == " ":

#user cannot enter whitespace or null

isAskedAddress = True

else:

print("Please enter a valid address.")

#making sure contact number is 10 digits long and non-string

isAskedContact = False

is10Numbered = False #assuming user didn't input 10 digits

while isAskedContact == False and is10Numbered == False:

returnContactNum = input("\nEnter your phone number: ")

isString = False

for chars in returnContactNum:

#converting each chars to lowercase to check ASCII value

if "a" <= chars.lower() and chars.lower() <= "z":

isString = True

break

if isString == True:

print("Enter non-string phone number.")

else:

#finally, only if non-string 10 digits are entered, the entered number is accepted

if len(returnContactNum) == 10:

is10Numbered = True

isAskedContact = True

else:

print("Enter 10 digits.")

returnAgain = True

isIdAvailable = True

while returnAgain == True:

try:

while isIdAvailable == True:

returnId = int(input("\nWhich land would you like to return? (Enter Kitta Id): "))

if returnId in landDict.keys():

if landDict[returnId][4] == " Not Available":

#changing availability in dictionary

landDict[returnId][4] = " Available"

isIdAvailable = False

else:

print("The land with that id is yet to be rented.")

else:

print("Our company doesn't have access to that land.")

#handeling ValueError of rent month

isAskedMonthrent = False

while isAskedMonthrent == False:

try:

rentMonth = int(input("\nHow many months did you rent it for? "))

isAskedMonthrent = True

except ValueError:

print("ValueError. Enter month in numbers.")

#handeling ValueError of return month

isAskedMonthreturn = False

while isAskedMonthreturn == False:

try:

returnMonth = int(input("\nHow many months has it been since you rented it? "))

isAskedMonthreturn = True

except ValueError:

print("ValueError. Enter month in numbers.")

totalPrice = ( int(landDict[returnId][3]) )\*returnMonth

countMonth = 1

totalPriceAfterFine = ( int(landDict[returnId][3]) )\*returnMonth #if not fined, totalPriceAfterFine = totalPrice

if returnMonth > rentMonth:

countMonth = returnMonth - rentMonth

totalPriceAfterFine = 0

totalPriceAfterFine = ( (int(landDict[returnId][3]))\*returnMonth ) + fineForContractBreachers(totalPrice, countMonth)

dict\_[returnId] = [landDict[returnId][0], landDict[returnId][1], landDict[returnId][2], landDict[returnId][3], str(returnMonth), str(totalPrice), str(totalPriceAfterFine)]

#asking the user if s/he wants to return rented land again

repeatReturn = True

while repeatReturn == True:

returnLandRepeat = input("\nWould you like to return a rented land again? (y/n): ")

if returnLandRepeat.lower() == "y":

#changing availability in text file so when list is printed, list is updated

changingAvailability(landDict)

#showing lands that the company controls in a list

print("\n")

print("="\*105)

print("\t\t\t\t\t\t Land List")

print("="\*105)

print("\nKitta Id \t City \t\tDirection Anna\t Price\t\t Availability")

file = open("land.txt", "r")

for line in file:

print("-"\*105)

#printing 105 \_

print(line.replace(",", "\t\t"))

#replacing , with double tab space and then printing file content

print("="\*105)

file.close()

returnAgain = True

repeatReturn = False

isIdAvailable = True

elif returnLandRepeat.lower() == "n":

returnAgain = False

repeatReturn = False

#calling bill generating function

print("\n\n")

returnBills(dict\_, returnUserName, returnContactNum, userAddress, returnMonth, rentMonth, countMonth)

changingAvailability(landDict) #changing availability in land.txt

continueAgain = True

while continueAgain == True:

continueOperation = input("\nDo you want to continue to return? (y/n): ")

if continueOperation.lower() == "y":

#showing lands that the company controls in a list

print("\n")

print("="\*105)

print("\t\t\t\t\t\t Land List")

print("="\*105)

print("\nKitta Id \t City \t\tDirection Anna\t Price\t\t Availability")

file = open("land.txt", "r")

for line in file:

print("-"\*105)

#printing 105 \_

print(line.replace(",", "\t\t"))

#replacing , with double tab space and then printing file content

print("="\*105)

file.close()

continueOperationBool = True

continueAgain = False

elif continueOperation.lower() == "n":

continueOperationBool = False

continueAgain = False

else:

print("Type option 'y' for yes and 'n' for no")

else:

print("Choose option 'y' for yes and 'n' for no")

except ValueError:

print("ValueError. Enter valid Kitta number from the above list.")

def fineForContractBreachers(totalPrice, countMonth):

"""

This function adds fine when the contract is breached, i.e., returnMonth>rentMonth

It has 2 parameters: totalPrice, countMonth

It returns the total price after fine is added

"""

totalPriceAfterFine = 0

totalPriceAfterFine += 0.1\*totalPrice\*countMonth

return totalPriceAfterFine

## write.py

import datetime

from read import readingFile

landDict = readingFile()

def currentDateTime():

"""

This function creates a unique filename based on current date and time and name of the user

"""

#current date and time is being taken and put into respective variables

year = str(datetime.datetime.now().year)

month = str(datetime.datetime.now().month)

day = str(datetime.datetime.now().day)

hour = str(datetime.datetime.now().hour)

minute = str(datetime.datetime.now().minute)

second = str(datetime.datetime.now().second)

dateTimeNow = year + "-" + month + "-" + day + "\_" + hour + "-" + minute + "-" + second

return dateTimeNow

currentDateTime()

def changingAvailability(landDict):

"""

Changes availablity of land on the file: land.txt

Parameter: landDict

"""

try:

file = open("land.txt", "w")

for key, value in landDict.items():

file.write(str(key)) #converting key to string explicitly

for item in value:

file.write("," + item) #joining the items in the value list with ", " separator

file.write("\n") #next line after each iteration

file.close()

except IOError as e:

#the exception object e captures information about the error. This information can be useful for debugging or providing more detailed

#error messages. In this case, specific error(e) is printed out.

print("Writing error to file:", e)

def rentBills(landDict, name, phn, address):

"""

This function generates a bill/invoice after the user rents land based on user's personal and rental information.

This function has parameters namely; landDict, name, phn, address

This function doesn't return anything.

"""

dateTimeNow = currentDateTime()

#a unique filename is generated using current time and user's name

fileName = dateTimeNow + "\_" + name + "\_" + "rented"+ ".txt"

#the file is opened and "Hello" user is written on it

file = open(fileName, "w")

print("="\*130)

file.write("="\*130 + "\n")

print("\t\t\t\t\t\t\t Techno Property Nepal\n\t\t\t\t\t\t\t Seti O.p. Marg 556\n\t\t\t\t\t\t\t Kathmandu 44600")

file.write("\t\t\t\t\t\t\t Techno Property Nepal\n\t\t\t\t\t\t\t Seti O.p. Marg 556\n\t\t\t\t\t\t\t Kathmandu 44600\n")

print("\t\t\t\t\t\t\t +01-5147400")

file.write("\t\t\t\t\t\t\t +01-5147400\n")

print("="\*130)

file.write("="\*130 + "\n")

print("\t\t\t\t\t\t\t\t\t\t\t\t\t\t Date: " + dateTimeNow.split("\_")[0])

file.write("\t\t\t\t\t\t\t\t\t\t\t\t\t\t Date: " + dateTimeNow.split("\_")[0] + "\n")

print("Bill Type: Rent")

file.write("Bill Type: Rent\n")

print("Billed To:" + name)

file.write("Billed To: " + name + "\n")

print("Provided Contact Info: " + phn)

file.write("Provided Contact Info: " + phn + "\n")

print("Provided Address: " + address + "\n\n")

file.write("Provided Address: " + address + "\n\n\n")

print("-"\*130)

file.write("-"\*130 + "\n")

print("Kitta Id \t City \t\t Direction \t\t Anna\t Price \t\t Duration \t Total Price")

print("-"\*130)

file.write("Kitta Id \t City \t\t Direction \t\t Anna\t Price \t\t Duration \t Total Price\n")

file.write("-"\*130 + "\n")

totalPrice = 0

for key, value in landDict.items():

print(str(key) + "\t\t" + value[0] + "\t " + value[1] + " \t\t " + value[2] + "\t\t" + value[3] + " \t\t " + value[4] + " \t\t" + value[5])

file.write(str(key) + "\t\t" + value[0] + "\t " + value[1] + " \t\t " + value[2] + "\t\t" + value[3] + " \t\t " + value[4] + " \t\t" + value[5] + "\n")

totalPrice += float(value[5])

print("-"\*130)

file.write("-"\*130 + "\n")

print("\n\nFinal Amount of all lands rented: ", str(totalPrice))

file.write("\n\nFinal Amount of all lands rented: " + str(totalPrice) + "\n")

print("="\*130)

file.write("="\*130 + "\n")

file.close()

def returnBills(landDict, name, phn, address, returnMonth, rentMonth, countMonth):

"""

This function generates a bill/invoice after user returns land based on user's personal and rental information.

This function has parameters namely; landDict, name, phn, address, returnMonth, rentMonth, countMonth.

This function doesn't return anything.

"""

dateTimeNow = currentDateTime()

#a unique filename is generated using current time and user's name

fileName = dateTimeNow + "\_" + name + "\_" + "returned"+ ".txt"

#the file is opened and "Hello" user is written on it

file = open(fileName, "w")

print("="\*166)

file.write("="\*166 + "\n")

print("\t\t\t\t\t\t\t\t\t Techno Property Nepal\n\t\t\t\t\t\t\t\t\t Seti O.p. Marg 556\n\t\t\t\t\t\t\t\t\t Kathmandu 44600")

file.write("\t\t\t\t\t\t\t\t\t Techno Property Nepal\n\t\t\t\t\t\t\t\t\t Seti O.p. Marg 556\n\t\t\t\t\t\t\t\t\t Kathmandu 44600\n")

print("\t\t\t\t\t\t\t\t\t +01-5147400")

file.write("\t\t\t\t\t\t\t\t\t +01-5147400\n")

print("="\*166)

file.write("="\*166 + "\n")

print("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t Date: " + dateTimeNow.split("\_")[0])

file.write("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t Date: " + dateTimeNow.split("\_")[0] + "\n")

print("Bill Type: Return")

file.write("Bill Type: Return\n")

print("Billed To:" + name)

file.write("Billed To: " + name + "\n")

print("Provided Contact Info: " + phn)

file.write("Provided Contact Info: " + phn + "\n")

print("Provided Address: " + address + "\n\n")

file.write("Provided Address: " + address + "\n\n\n")

print("-"\*166)

file.write("-"\*166 + "\n")

print("Kitta Id \t City \t\t Direction \t Anna\t Price \t Rented For\tReturned At \t Total Price \t Total Price After Fine")

print("-"\*166)

file.write("Kitta Id \t City \t\t Direction \t Anna\t Price \t\t Rented For \t Returned At \t Total Price \t Total Price After Fine\n")

file.write("-"\*166 + "\n")

totalAmountAfterFine = 0

totalAmount = 0

for key, value in landDict.items():

print(str(key) + "\t\t" + value[0] + "\t " + value[1] + "\t\t" + value[2] + "\t " + value[3] + " \t\t " + str(rentMonth) + "\t\t " + value[4] + " \t\t " + value[5] + "\t\t " + value[6])

file.write(str(key) + "\t\t" + value[0] + "\t " + value[1] + "\t\t" + value[2] + "\t " + value[3] + " \t\t " + str(rentMonth) + "\t\t " + value[4] + " \t\t " + value[5] + "\t\t " + value[6] + "\n")

totalAmount += float(value[5])

totalAmountAfterFine += float(value[6])

print("-"\*166)

file.write("-"\*166 + "\n")

print("\n\nTotal Amount: ", str(totalAmount))

file.write("\n\nTotal Amount: " + str(totalAmount) + "\n")

print("Final Amount (after fine): ", str(totalAmountAfterFine))

file.write("Final Amount (after fine): " + str(totalAmountAfterFine) + "\n")

print("="\*166)

file.write("="\*166 + "\n")

file.write("\n")

file.close()

## main.py

from operations import renting

from operations import returning

isRunning = True

dict\_ = {} #dictionary for storing rented values

while isRunning == True: #running the loop until loopRunning is false

try:

#showing lands that the company controls in a list

print("\n")

print("="\*105)

print("\t\t\t\t\t\t Land List")

print("="\*105)

print("\nKitta Id \t City \t\tDirection Anna\t Price\t\t Availability")

file = open("land.txt", "r")

for line in file:

print("-"\*105)

#printing 105 \_

print(line.replace(",", "\t\t"))

#replacing , with double tab space and then printing file content

print("="\*105)

file.close()

print("\n\nWelcome To Land Rental.")

print("What would you like to do?")

print("1 - Rent a land")

print("2 - Return a rented land")

print("3 - Exit")

userChoice = int(input("\nEnter your choice: "))

#if user chooses option 1, land renting code runs

if userChoice == 1:

renting()

print("Thank you for choosing our service.\n\n\n")

#if user chooses option 2, land returning code runs

elif userChoice == 2:

returning()

print("\nThank you for choosing our service.\n\n\n")

#if user chooses option 3, loop is terminated

elif userChoice == 3:

print("Come back again!!")

isRunning = False

else:

print("\nInvalid Input! Try again with available numbers.\n\n")

#if numeric value is entered, the user is asked to try again

except ValueError:

print("ValueError Found!! Try using available numbers to go forward with the operation.\n\n")