



**CS4051NI Fundamentals of Computing**

**60% Individual Coursework**

**2023/24 Spring**

**Student Name: Aayush Shrestha**

**London Met ID: 23050351**

**College ID: NP01CP4A230223**

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

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

**Word Count: 11703**

**Project File Links:**

|  |  |
| --- | --- |
| **YouTube Link:** | Keep Unlisted YouTube URL of your Project Here |
| **Google Drive Link:** | Keep Google Drive URL of your Project Here with Anyone in Organization can View Option Enabled |

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

Table of Contents

[1. Introduction 1](#_Toc165919616)

[1.1 Goals and objective 1](#_Toc165919617)

[2. Discussion and analysis 2](#_Toc165919618)

[2.1 Algorithm 2](#_Toc165919619)

[2.2 Flowchart 4](#_Toc165919620)

[2.3 Pseudocode 5](#_Toc165919621)

[2.3.1 Pseudocode of main.py 5](#_Toc165919622)

[2.3.2 Pseudocode of Operation.py 6](#_Toc165919623)

[2.3.3 Pseudocode of read.py 14](#_Toc165919624)

[2.3.4 PSEUDOCODE of write.py 17](#_Toc165919625)

[2.4 Data Structures 23](#_Toc165919626)

[2.4.1 Lists 23](#_Toc165919627)

[2.4.2 Dictonary 24](#_Toc165919628)

[2.4.4 Sets 24](#_Toc165919629)

[2.4.5 Int 25](#_Toc165919630)

[2.4.5 Float 25](#_Toc165919631)

[2.4.6 String 25](#_Toc165919632)

[2.4.7 Boolean 26](#_Toc165919633)

[3. Program 26](#_Toc165919634)

[3.1 Implementation of program 26](#_Toc165919635)

[3.2 Renting and Returning the land 27](#_Toc165919636)

[4. Testing 36](#_Toc165919637)

[4.1 Test – 1: To show implementation of try, except 36](#_Toc165919638)

[4.2 Test – 2: To select rent and return of lands 38](#_Toc165919639)

[4.3 Test – 3: To generate file of renting of land(s) 41](#_Toc165919640)

[4.4 Test – 4: To generate file of returning the land(s) 48](#_Toc165919641)

[4.5 Test – 5: Showing the update in stock of land(s) 56](#_Toc165919642)

[5. Conclusion 65](#_Toc165919643)

[6. References 66](#_Toc165919644)

[7. Appendix 67](#_Toc165919645)

[7.1 Main.py 67](#_Toc165919646)

[7.2 Operation.py 68](#_Toc165919647)

[7.3 Write.py 72](#_Toc165919648)

[7.4 Read.py 77](#_Toc165919649)

[7.5 TechnoPropertyNepal.txt 79](#_Toc165919650)

**List of Figures**

[Figure 1: Flow chart of the program 4](#_Toc165919542)

[Figure 2: Use of list in my program 24](#_Toc165919543)

[Figure 3: Use of int in my program 25](#_Toc165919544)

[Figure 4: Use of string in my program 26](#_Toc165919545)

[Figure 5: Use of Boolean in my program 26](#_Toc165919546)

[Figure 6: Renting and returning the land – 1 28](#_Toc165919547)

[Figure 7: Renting and returning the land – 2 28](#_Toc165919548)

[Figure 8: Renting and returning the land – 3 29](#_Toc165919549)

[Figure 9: Renting and returning the land – 4 29](#_Toc165919550)

[Figure 10: Renting and returning the land – 5 30](#_Toc165919551)

[Figure 11: Renting and returning the land – 6 30](#_Toc165919552)

[Figure 12 :Renting and returning the land – 7 31](#_Toc165919553)

[Figure 13: Renting and returning the land – 8 31](#_Toc165919554)

[Figure 14: Renting and returning the land – 9 32](#_Toc165919555)

[Figure 15: Renting and returning the land – 10 32](#_Toc165919556)

[Figure 16: Renting and returning the land – 11 33](#_Toc165919557)

[Figure 17: Renting and returning the land – 12 33](#_Toc165919558)

[Figure 18: Renting and returning land – 13 34](#_Toc165919559)

[Figure 19: Renting and returning land – 14 34](#_Toc165919560)

[Figure 20: Renting and returning land – 15 35](#_Toc165919561)

[Figure 21: Renting and returning land - 16 35](#_Toc165919562)

[Figure 22: Renting and returning the land – 19 36](#_Toc165919563)

[Figure 23: Test -1 – Choosing 1 in main menu 37](#_Toc165919564)

[Figure 24: Test -1 - Giving name as input 37](#_Toc165919565)

[Figure 25: Test -1 – Giving a as input in input kitta 38](#_Toc165919566)

[Figure 26: Test -2: To select rent and return of lands - 1 39](#_Toc165919567)

[Figure 27: Test -2: To select rent and return of lands - 2 39](#_Toc165919568)

[Figure 28: Test -2: To select rent and return of lands - 3 40](#_Toc165919569)

[Figure 29: Test -2: To select rent and return of lands - 4 40](#_Toc165919570)

[Figure 30: Test -2: To select rent and return of lands - 5 41](#_Toc165919571)

[Figure 31: Test -2: To select rent and return of lands - 6 41](#_Toc165919572)

[Figure 32: Test – 3 – Homescreen and giving 1 as input 42](#_Toc165919573)

[Figure 33: Test – 3 - Giving Hari Shrestha as input in name 43](#_Toc165919574)

[Figure 34: Test – 3 - Giving the kitta number 43](#_Toc165919575)

[Figure 35: Test – 3: - Giving the months rented 44](#_Toc165919576)

[Figure 36: Test – 3 - Giving the rent another land yes 44](#_Toc165919577)

[Figure 37: Test – 3 - Giving kitta number 101 as input in kitta 45](#_Toc165919578)

[Figure 38: Test – 3 - Giving month as input 45](#_Toc165919579)

[Figure 39: Test – 3 - Giving no as input in rent more and invoice generation 46](#_Toc165919580)

[Figure 40: Test – 3 - Invoice generated location 46](#_Toc165919581)

[Figure 41: Test – 3 - Renting invoice 47](#_Toc165919582)

[Figure 42: Test – 3 - Updated lands in home screen 47](#_Toc165919583)

[Figure 43: Test – 3 - Updated lands text file 48](#_Toc165919584)

[Figure 44: Test – 4 - Home screen and giving 2 as input 49](#_Toc165919585)

[Figure 45: Test – 4 – Giving name as input 49](#_Toc165919586)

[Figure 46: Test – 4 - Giving kitta numbers as input 50](#_Toc165919587)

[Figure 47: Test – 4 - Giving month rented as input 50](#_Toc165919588)

[Figure 48: Test – 4 - Giving returned month as input 51](#_Toc165919589)

[Figure 49: Test – 4 - Giving yes as input to return more land 51](#_Toc165919590)

[Figure 50: Test – 4 - Giving second kitta number as input 52](#_Toc165919591)

[Figure 51: Test – 4 - Giving no of months rented 52](#_Toc165919592)

[Figure 52: Test – 4 - Giving returned month as input 53](#_Toc165919593)

[Figure 53: Test – 4 - Giving no as input to return more land 53](#_Toc165919594)

[Figure 54: Test – 4 - Generating and displaying invoice 54](#_Toc165919595)

[Figure 55: Test – 4 - Updated land availability in the cell 54](#_Toc165919596)

[Figure 56: Test – 4 - Generation of return invoice in the folder 55](#_Toc165919597)

[Figure 57: Test – 4 - Return invoice 55](#_Toc165919598)

[Figure 58: Test – 4 - Updated lands in land file 56](#_Toc165919599)

[Figure 59: Test – 5 - Home screen and typing 1 as input 57](#_Toc165919600)

[Figure 60: Test – 5 - Giving name as input 58](#_Toc165919601)

[Figure 61: Test – 5 - Giving kitta number as input 58](#_Toc165919602)

[Figure 62: Test – 5 - Giving month's rented as input 59](#_Toc165919603)

[Figure 63: Test – 5 - Giving no as input in rent another land 59](#_Toc165919604)

[Figure 64: Test – 5 - Generating invoice and displaying the invoice 60](#_Toc165919605)

[Figure 65: Test – 5 - Updated availability of the land in the home screen 60](#_Toc165919606)

[Figure 66: Test – 5 - Land file updated by the program 61](#_Toc165919607)

[Figure 67: Test – 5 - Giving 2 in option and giving full name 61](#_Toc165919608)

[Figure 68: Test – 5 – Giving the kitta number of the land to return 62](#_Toc165919609)

[Figure 69: Test – 5 – Giving the number of months rented 62](#_Toc165919610)

[Figure 70: Test – 5 - Giving the months returned 63](#_Toc165919611)

[Figure 71: Test – 5 - Giving no as input in return again 63](#_Toc165919612)

[Figure 72: Test – 5 - Generated invoice and displayed on screen 64](#_Toc165919613)

[Figure 73: Test – 5 - Availability status changed to available after return 64](#_Toc165919614)

[Figure 74: Test – 5 - Availability status changed in lands file 65](#_Toc165919615)

**Table of tables**

[Table 1: Test -1: To show Implementation of try, except 37](#_Toc165919651)

[Table 2: Test - 2: To select rent and return of lands 38](#_Toc165919652)

[Table 3: Test – 3: To generate file of renting of lands 42](#_Toc165919653)

[Table 4: Test – 4: To generate file of returning the land(s) 48](#_Toc165919654)

[Table 5: Test – 5: Showing the update in stock of land(s) 57](#_Toc165919655)

# 1. Introduction

Python is an interpretive, object-oriented and high-level programming language. It is one of the most easy and practical programming languages. Python is mostly used in software development sector, web development and competitive programming. For those who are new in learning programming and software engineering, it is advised because of its simple syntax. Its popularity is increasing rapidly as it has numerous applications in modern technologies such as data science, machine learning and automation task. It has been repeatedly recognized as one of the best programming languages for many years. Now, python is utilized in many different types of development, ranging from simple scripting, game creation and fundamental programming to large scale complex software development. Python boasts an extensive library and robust community, with various frameworks available for both frontend and backend web development (GeeksforGeeks, 2024).

In this project, we were assigned to make a program for a company called Techno Property Nepal. The company has different lands in different locations with different prices. We were to make a program so that the user who uses the program can easily go through the lands in the system of techno property company and easily rent and return the lands available in the system. When renting and returning the land, the program should also create a unique invoice which includes all the lands rented or returned with the user’s details and total prices of all the lands rented or returned.

## Goals and objective

The goal of this project is to make a simple application that will manage the lands of a company named Techno Property by providing a simple application to rent and return land. The objectives of this coursework are:

To teach us the usage of functions, lists and different types of data’s

To teach us modular programming in python

To make us know how to handle errors

To teach us good practice in coding in python

To teach us how to do file handling

# 2. Discussion and analysis

## 2.1 Algorithm

An algorithm is the process for carrying out a calculation or problem-solving step by step. In hardware or software-based routines, algorithms function as a precise set of instructions that carry out predetermined operations one after another. All branches of information technology heavily rely on algorithms. An algorithm typically is used in computer science and computer programming to describe a brief process that resolves an issue. Algorithm is essential to automate system because they provide guidelines for processing data (Gillis, 2023).

Algorithm for the project are as follows:

**Step 1:** Start

**Step 2:** Print all the lands in the system.

**Step 3:** Ask the user to input from 1-3, 1 for renting the land, 2 for returning the land and

3 for exiting the program

**Step 4:** If the user input is 1 then display all the available lands and go to step 5, if the

user input is 2 then go to Step 17, if the user input is 3 go to Step 30, else ask the

user to give a valid input

**Step 5:** Ask the user for their name

**Step 6:** Ask the user for the kitta number of the land they want to rent

**Step 7:** Check whether the user given kitta number is available in the system.

**Step 8:** If the kitta is not available or not in the system then display the land is not available

and go to Step 6, else go to Step 9

**Step 9:** Ask the user for the number of months they want to rent

**St7ep 10:** If the months is greater than 12, go to Step 11, else go to Step 12

**Step 11:** Print the user cannot rent the land more than 12 months at once then go to Step

9

**Step 12:** Change the status of the land available to not available.

**Step 13:** Ask the user if they want to rent another land, if the answer is yes, go to Step 6,

else if the answer is no then go to Step 15, else go to Step 14.

**Step 14:** Print that the value is invalid and go to Step 13

**Step 15:** Calculate the total price and generate invoice.

**Step 16:** Display the invoice to the user and return Step 2

**Step 17:** Ask the user for their name.

**Step 18:** Ask the user for the kitta number they want to return.

**Step 19:** Check whether the kitta number is in the system, if the kitta number is in the

system is returned or not, if the kitta number is returned go to step 20, else go to

Step 21.

**Step 20:** Display that the land is not in the system or already returned, then go to Step

18

**Step 21:** Ask the number of months they rented the land

**Step 22:** Ask the number of month after they are returning the land

**Step 23:** If the returned month is greater than months returned, then go to Step 24, else

If the returned month is smaller than the months rented then go to Step 25, else

go to Step 26.

**Step 24:** Charge the monthly price and charge 10% more on the number of months extra

and calculate the total price, then go to Step 27.

**Step 25:** Charge the monthly price till the returned month and don’t charge extra and

Calculate the total price, then go to Step 27.

**Step 26:** Charge the monthly price according to the months rented with no extra charge

and calculate the total price, then go to Step 27.

**Step 27:** Ask the user if they want to return another land, If the answer is yes, go to Step

18, else if the answer is no then go to Step 29, else go to Step 28.

**Step 28:** Display a message saying to give a valid input then go to step 27.

**Step 29:** Generate the invoice and display the user, then go to Step 1.

**Step 30:** Stop the program

## 2.2 Flowchart

Flowcharts are simply graphical representations of algorithms that help user comprehend the code graphically. It presents step by step solutions to the problem. It is a visual representation of the program, that most programmers do at the beginning of the process to help better understand the algorithms and computer science. It also helps in troubleshooting algorithm issues. A flowchart consists of boxes that shows the sequential process of solving a problem. A flow chart makes a process or algorithm easy to comprehend since it’s a visual depiction of the whole process caried out in the program (GeeksforGeeks, 2023).

The flow chart for the project are as follows:

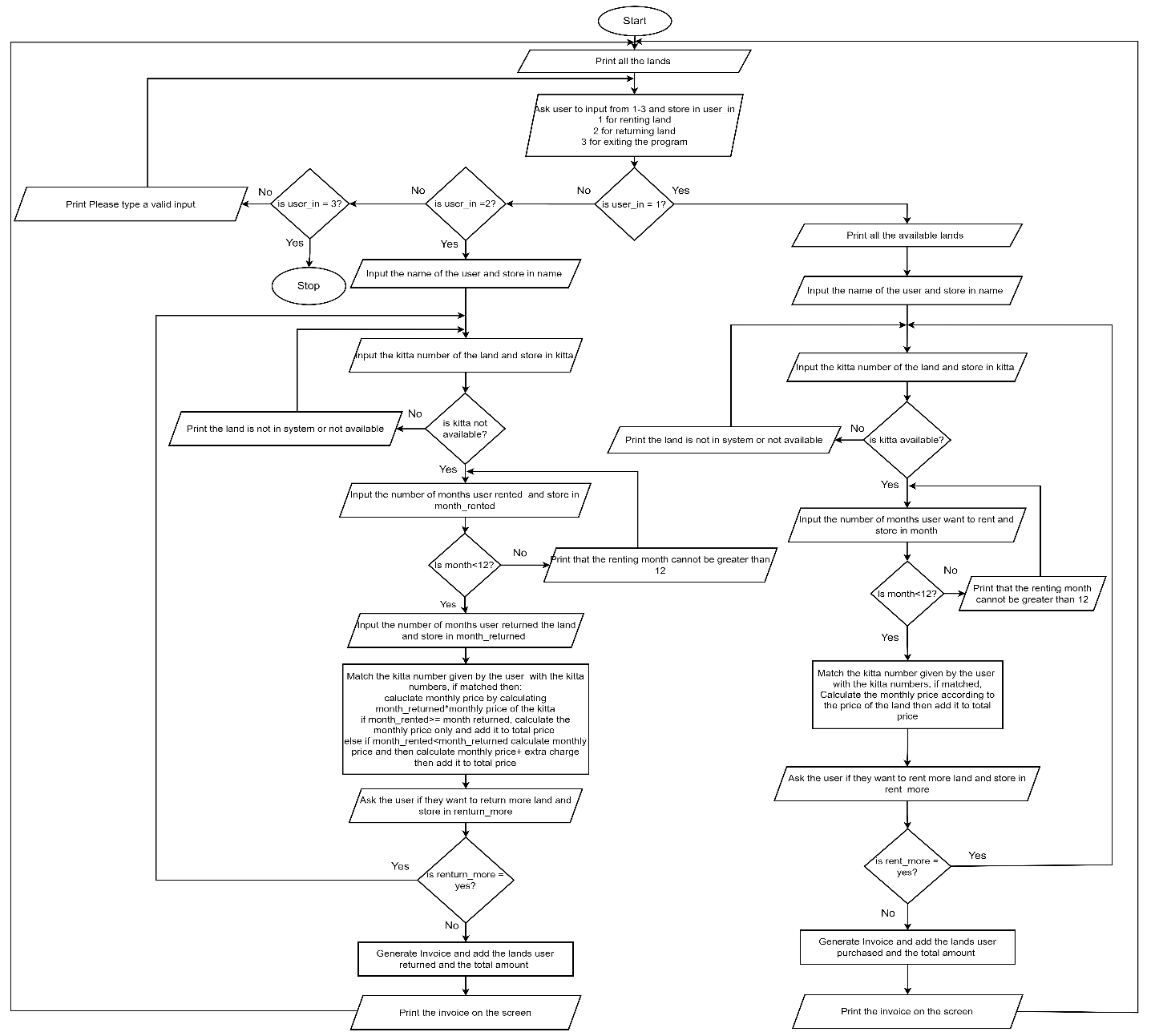


Figure 1: Flow chart of the program

## 2.3 Pseudocode

Pseudocode is a comprehensive, clear explanation of the intended actions of an algorithm or computer program. To make it easy to understand for the programmers and other developers, it is written in a formal yet understandable language with natural grammar and structure. It is not possible to convert pseudocode into an executable program as it is not a programming language. Rather, it acts as a guide for converting the logic of the code into a programming language. As it is written in understandable format, the team can inspect pseudo code ensuring that the program will stick to design criteria. Errors are easier to spot and fix in pseudocode stage. The same pseudocode can be converted into different programming languages (Sheldon, 2023).

### 2.3.1 Pseudocode of main.py

**IMPORT** necessary modules (read, operation, write)

**SET** the loop to True

**WHILE** loop is True:

**DO**

**DISPLAY** all the available lands by calling the function all\_lands from read module

**TRY:**

**DO**

**DISPLAY** the choices available by calling the function available\_lands() from read module

**GET** the users input and store it in userin

**IF** the value of userin is 1:

**DO**

**Display** all the available lands in the system

**Call** the function rent\_lands() from the operation module

**END DO**

**ELSE IF** the value of userin is 2:

**DO**

**Call** the function return\_lands() from the operation module

**END DO**

**Else If** the value of the userin in 3:

**DO**

**SET** the value of loop to False

**END DO**

**ELSE:**

**DO**

**DISPLAY** to type a valid number from 1-3

**END DO**

**END IF**

**END DO**

**EXECPT:**

**DO**

**PRINT** that the input is invalid and to type a number

**END DO**

**END DO**

### 2.3.2 Pseudocode of Operation.py

**IMPORT** necessary modules(read,write)

**DEFINE** a function input\_month\_rented():

**DO**

**SET** the month\_check to **True**

**LOOP** while month\_check is **true**:

**DO**

**TRY**:

**DO**

**INPUT** the month rented from the user and store it in month

**IF** month is greater than 12 or month is less than or equal to 0  
 **DISPLAY** the input is invalid and months must be between 1-

12

**ENDDO**   
 **ELSE:**

**DO**

**SET** the month\_check to **False**

**Return** the value of month

**END DO**

**END IF**

**END DO**

**EXCEPT**

**DISPLAY** the input is invalid and ask the user to try again

**END DO**

**END DO**

**END FOR**

**END DO**

**DEFINE** a function named input\_month\_returned without parameters:

**DO**

**SET** the variable check to **True**

**LOOP** while check is **True:**

**DO**

**TRY**

**DO**

**INPUT** the month returned from the user and store it in month\_returned

**IF** month\_returned is greater than 0:

**DO**

**SET** the check to **False**

**RETURN** the value of month\_returned

**END DO**

**ELSE**

**DO**

**DISPLAY** that the value is invalid and to provide a positive

**END DO**

**END IF**

**END DO**

**EXCEPT**

**DO**

**DISPLAY** the given value is not valid and try again

**END DO**

**END DO**

**END WHILE**

**END DO**

**DEFINE** a function called input\_kitta\_rent without any parameters

**DO**

**SET** the variable check to **True**

**LOOP** while the check variable is **True**

**DO**

**TRY**

**DO**

**INPUT** kitta number of the land they want to rent and store in kitta

**CALL** the function check\_kitta from read module and store the

returned value in check\_kitta

**IF** the value of check\_kitta is **True:**

**DO**

**SET** the value of variable check to **False**

**RETURN** the value of variable kitta

**END DO**

**ELSE**

**DO**

**DISPLAY** the kitta is not available in the system and to try again

**END DO**

**END IF**

**END DO**

**EXCEPT**

**DO**

**DISPLAY** the value is invalid and to try again

**END DO**

**END DO**

**END DO**

**END WHILE**

**END DO**

**DEFINE** a function named input\_kitta\_return which takes no parameters:

**DO**

**SET** the value of variable check to **True**

**LOOP** while check is true

**DO**

**TRY**

**DO**

**INPUT** kitta number of the month they want to return and store in a

variable named kitta

**CALL** the function return\_check\_kitta and pass the variable kitta in

the function, store the returned value in check\_return\_kitta

**IF** the value of check\_return\_kitta is true

**DO**

**SET** the value of check to **False**

**RETURN** the value of kitta

**END DO**

**ELSE**

**DO**

**DISPLAY** the kitta number is returned or is not in system

**END DO**

**END IF**

**END DO**

**EXEPT**

**DO**

**DISPLAY** the input is invalid and try again

**END DO**

**END DO**

**END WHILE**

**END DO**

**DEFINE** a function rent\_lands without any parameter

**DO**

**DECLARE**  and set the variable rent\_land to **True**

**DECLARE** and set the variable kitta\_available to **True**

**DECLARE** and set the variable rent to **True**

**DECLARE** a list variable kittas to store kittas

**DECLARE** a list variable months to store the months

**INPUT** full name from the user and store it in a variable name

**LOOP** **while** rent\_land is **True:**

**DO**

**LOOP** while kitta\_available is **True**

**DO**

**CALL** the function input\_kitta\_rent and store the value in kitta

**CALL** the function input\_month\_rented and store the value in the variable month

**APPEND** the value of kitta in kittas list

**APPEND** the value of month in months list

**CALL** the function update\_rent\_lands and pass the variable kitta as parameter to update the rented land

**DECLARE** a variable rent and set its value to **True**

**LOOP** while the value of rent is **True**

**DO**

**INPUT** if the user wants to rent another land and lower the

lower the input and store it in a variable rent\_more

**IF** rent\_more equals yes or rent\_more equals y then

**DO**

**SET** the value of kitta\_available to **True**

**SET** the value of rent\_land to **True**

**SET** the value of rent to **False**

**END DO**

**ELSE IF** the value of rent\_more is no

**DO**

**SET** the value of kitta\_available to **False**

**SET** the value of rent\_land to **False**

**SET** the value of rent to **False**

**DISPLAY** that the invoice is generating

**CALL** the function rent\_invoice from the write module

and pass the variable name, kittas and months as parameter

**ASK** the user to input Enter key to go to home screen

**END** **DO**

**ELSE:**

**DO**

**DISPLAY** to type a valid input and try again

**END** **DO**

**END IF**

**END DO**

**END DO**

**END WHILE**

**END DO**

**END WHILE**

**END DO**

**DEFINE** a function return\_lands without any parameter

**DO**

**DECLARE** a list variable months\_rented to store the months rented

**DECLARE** a list variable months\_returned to store returned months

**DECLARE** a list variable kittas to store kittas

**DECLARE** and set the variable return\_land to **True**

**DECLARE** and set the variable return\_more to **True**

**INPUT** full name from the user and store it in a variable name

**LOOP** while return\_land is **True**:

**DO**

**LOOP** while kitta\_available is **True**

**DO**

**CALL** the function input\_kitta\_return and store the value in kitta

**CALL** the function input\_month\_rented and store the value in the variable month\_rented

**CALL** the function input\_month\_returned and store the value in the variable month\_returned

**APPEND** the value of kitta in kittas list

**APPEND** the value of month\_rented in months\_rented list

**APPEND** the value of month\_returned in months\_returned list

**CALL** the function update\_returned\_kitta and pass the variable kitta as parameter to update the returned land

**DECLARE** a variable return\_more and set its value to **True**

**LOOP** while the value of return\_more is **True**

**DO**

**INPUT** if the user want to return another land and lower the

lower the input and store it in a variable return\_more

**IF** return\_more equals yes or return\_more equals y then

**DO**

**SET** the value of return\_land to **True**

**SET** the value of return\_more to **False**

**END** **DO**

**ELSE IF** the value of return\_more is no or the value of return\_more is n then

**DO**

**SET** the value of return\_land to **False**

**SET** the value of return\_more to **False**

**DISPLAY** that the invoice is generating

**CALL** the function return\_invoice from the write module and pass the variable name, kittas, months\_rented and months\_returned as parameter

**ASK** the user to input Enter key to go to home screen

**END** **DO**

**ELSE:**

**DO**

**DISPLAY** to type a valid input and try again

**END DO**

**END IF**

**END DO**

**END WHILE**

**END DO**

**END WHILE**

**END DO**

**END WHILE**

**END DO**

### 2.3.3 Pseudocode of read.py

**DEFINE** a function named all\_lands without any parameter

**DO**

**OPEN** the TechnoPropertyNepal file in read mode

**DISPLAY** Kitta number , Location, Direction, Aana , Price, Availability with space

**FOR** each line in tie file

**DO**

**REPLACE** the \n with a blank space and split the words after ‘,’

` **DISPLAY** the land details line by line with proper spacing

**END** **DO**

**END FOR**

**CLOSE** the file

**END** **DO**

**DEFINE** a function available\_lands which doesn’t take any parameters

**DO**

**OPEN** the TechnoPropertyNepal file in read mode

**DISPLAY** Kitta number, Location, Direction, Aana, Price, Availability with space

**FOR** each line in file

**DO**

**REPLACE** the \n with a blank space and split the words after ‘,’

**IF** the availability of the land in the line is equals to available

**DO**

**PRINT** each words in line with appropriate space

**END** **DO**

**END IF**

**END DO**

**END FOR**

**CLOSE** the file

**END DO**

**DEFINE** a function named available\_lands which does not take any parameters:

**DO**

**OPEN** the file named TechnoPropertyNepal.txt in read mode

**DISPLAY** Kitta number, Location, Direction, Aana, Price, Availability with space

**FOR** each line in file:

**DO**

**REPLACE** the \n with a blank space and split the words in ‘,’

**IF** the availability of the line is available

**DO**

**DISPLAY** the land details on the screen

**END DO**

**END IF**

**END DO**

**END FOR**

**CLOSE** the file

**END DO**

**DEFINE** a function named check\_kitta and pass a parameter named kitta

**DO**

**OPEN** the file TechnoPropertyNepal in read mode

**FOR** each line in the file

**DO**

**REPLACE** the \n in line with a blank space and split the word where there is ‘,’

**IF** the availability of the line is available and if the kitta given by user matches the lines kitta

**DO**

**RETURN** the value **True**

**END DO**

**END IF**

**END DO**

**END FOR**

**CLOSE** the file

**END** **DO**

**DEFINE** a function named return\_check\_kitta which takes in kitta as parameter

**DO**

**OPEN** the file TechnoPropertyNepal in read more

**FOR** each line in file

**DO**

**REPLACE** the ‘\n’ in line and split the line into parts where there is ‘,’

**IF** the kitta number from parameter matches the kitta number of line and if

the availability of the land is ‘no available’

**DO**

**RETURN** the value true

**END** **DO**

**END IF**

**END DO**

**END FOR**

**CLOSE** the file

**END** **DO**

**DEFINE** a function read\_lands without any parameter

**DO**

**OPEN** the file TechnoPropertyNepal in read mode

**DECLARE** a list named lands

**FOR** each line in the files

**DO**

**REPLACE** the ‘\n’ in line with ‘’ and split the line in words where there is ‘,’

**APPEND** the line in the land list

**END DO**

**END FOR**

**RETURN** the list land

**END** **DO**

### 2.3.4 PSEUDOCODE of write.py

**IMPORT** datetime from datetime

**IMPORT** the module read

**DEFINE** a function update\_rent\_lands and pass a parameter named kitta

**DO**

**CALL** the function read\_lands from read module and set the returned value in a variable lines

**OPEN** the file TechnoPropertyNepal in write mode

**FOR** each line in lines

**DO**

**Split** the line where there is ‘,’ , strip the line and add the value to a v variable named parts

**If** the value of kitta number in variable parts is equal to the kitta number in

the variable kitta

**DO**

**DECLARE** a variable update\_line and add all the words in part and

separate it with ‘,’ and make the availability to not available

**WRITE** the variable update\_line in the file

**END DO**

**ELSE**

**DO**

**WRITE** the line in the file without changing

**END DO**

**END IF**

**END DO**

**END FOR**

**CLOSE** the file

**END DO**

**DEFINE** a function named rent\_invoice which takes in name,kittas,months as parameters

**DO**

**CALL** the method datetime.now from datetime and store in variable now

**DECLARE** a variable current\_time and set its value to the string of the current

time in the format hour minute and second

**DECLARE** and set the variable total\_price to 0

**DECLARE** and set the variable total\_price\_display to 0

**DECLARE** the variable invoice\_name and set its value to Invoice\_Rent\_ + the

value of variable name

**FOR** each i in kittas:

**DO**

**ADD** the kitta number to the variable invoice\_name separated by “\_”

**END** **DO**

**ADD** the current time and .txt to invoice\_name

**OPEN** a file named invoice\_name with write mode as invoice\_file

**WRITE** “-” in invoice\_file times 98

**WRITE** white space times 33 and Techno Prpoerty Nepal Rental Invoice in invoice\_file

**WRITE** Renters name followed by appropriate white spacing and then current

Date and time in invoice\_file

**Write** Kitta number, location, direction, Aana, Price, Months Rented, Monthly Total in invoice\_file

**WRITE** “-“ times 98

**FOR** variable i in range of length of kittas list

**DO**

**FOR** each lines in the file

**DO**

**REPLACE** the ‘\n’ with blank space and split the words where there is ‘,’ and store it in list named line

**SET** the variable monthly\_total to 0

**IF** the kitta number of line matches the kitta number in variable kittas with index i

**DO**

**MULTIPLY** monthly price from line with month list with index i and add it to monthly\_total

**WRITE** all the details of line with appropriate spacing and

also the months having index i and monthly total in invoice\_file

**WRITE** “-“ times 98 in invoice\_file

**ADD** the monthly\_total to total price

**END DO**

**END IF**

**END DO**

**END FOR**

**END DO**

**END FOR**

**WRITE** the value of total\_price in invoice\_file

**CLOSE** the invoice\_file

**DISPLAY** “-” times 98

**DISPLAY** white space times 33 and Techno Prpoerty Nepal Rental Invoice

**DISPLAY** “-“ times 98

**DISPLAY** Renters name followed by appropriate white spacing and then current

Date and time in invoice\_file

**DISPLAY** Kitta number, location, direction, Aana, Price, Months Rented, Monthly total

**DISPLAY** “-“ times 98

**CALL** the function read\_lands form read module and store the returned value in lands

**For** index i in kittas

**DO**

**FOR** each line in lands

**DO**

**Replace** the ‘\n’ in line and split where there is ‘,’ then store it in variable lines

**SET** the variable monthly\_total to 0

**IF** the kitta of lines matches the kitta of list kittas with index i

**DO**

**MULTIPLY** lines monthly price with the value in list months with index I and store it in variable monthly\_total

**DISPLAY** all the details of line with appropriate spacing and

also the months having index i and monthly total

**DISPLAY** “-” times 98

**ADD** the monthly\_total to total\_price\_display

**END DO**

**END IF**

**END DO**

**END FOR**

**END DO**

**END FOR**

**DISPLAY** the variable total\_price\_display

**END** **DO**

**DEFINE** a function update\_returned\_kitta and pass a parameter named kitta

**DO**

**CALL** the function read\_lands from read module and set the returned value in a variable lines

**OPEN** the file TechnoPropertyNepal in write mode

**FOR** each line in lines

**DO**

**Split** the line where there is ‘,’ , strip the line and add the value to a variable named parts

**If** the value of kitta number in variable parts is equal to the kitta number in

the variable kitt

**DO**

**DECLARE** a variable update\_line and add all the words in part and

separate it with ‘,’ and make the availability to Available

**WRITE** the variable update\_line in the file

**END DO**

**ELSE**

**DO**

**WRITE** the line in the file without changing anything

**END DO**

**END IF**

**END DO**

**END FOR**

**CLOSE** the file

**END DO**

**DEFINE** a function named return\_invoice which takes in name, kittas, months\_rented, months\_returned as parameters

**DO**

**CALL** the method datetime.now from datetime and store in variable now

**DECLARE** a variable current\_time and set its value to the string of the current

time in the format hour minute and second

**DECLARE** and set the variable total\_price to 0

**DECLARE** and set the variable total\_price\_display to 0

**DECLARE** and set the variable remaining\_month to 0

**DECLARE** the variable invoice\_name and set its value to Invoice\_Rent\_ with the

value of variable name

**FOR** each i in kittas:

**DO**

**ADD** the string i to the variable invoice\_name separated by “\_”

**END** **DO**

**ADD** the current time and .txt to invoice\_name

**OPEN** a file named invoice\_name with write mode as invoice\_file

**WRITE** “-” in invoice\_file times 150

**WRITE** white space times 59 and Techno Property Nepal Return Invoice in invoice\_file

**WRITE** Renters name followed by appropriate white spacing and then current

Date and time in invoice\_file

**WRITE** Kitta number, location, direction, Aana, Price, Months Rented, Months Returned, Monthly Rent and Extra Charge in invoice\_file

**WRITE** “-“ times 150

**FOR** variable i in range of length of kittas list

**DO**

**FOR** each lines in the lands file

**DO**

**REPLACE** the ‘\n’ with blank space and split the words where there is ‘,’ and store it in list named line

**SET** the variable monthly\_price to 0

**SET** the variable extra\_charge to 0

**IF** the kitta number of line matches the kitta number in variable kittas with index i

**DO**

**MULTIPLY** monthly price from line with month list with index i and add it to monthly\_price

**IF** the months\_rented with index i is greater than months\_returned with index I or the months\_rented with index i equals months\_returned with index i

**DO**

**ADD** the monthly\_price to total\_price

**END** **DO**

**ELSE IF** the months\_rented with index I is lsss than months\_returned with index i

**DO**

**SUBTRACT** the months\_rented with index I from months\_returned with index I and assign the value to the variable remaining\_month

**MULTIPLY** the value of monthly price of line with 10% and then multiply the value with remaining\_month , then assign the value to extra\_charge

**Add** the monthly\_price and extra charge, then add it to the total\_price

**END DO**

**END IF**

**WRITE** all the details of line with appropriate spacing and

also the months having index i and monthly total in invoice\_file

**WRITE** “-“ times 150 in invoice\_file

**END DO**

**END IF**

**END DO**

**END FOR**

**END DO**

**END FOR**

**WRITE** total\_price in the invoice\_file

**CLOSE** the invoice\_file

**DISPLAY** “-” times 130

**DISPLAY** white space times 46 and Techno Prpoerty Nepal Rental Invoice

**DISPLAY** “-“ times 130

**DISPLAY** Renters name followed by appropriate white spacing and then current

Date and time in invoice\_file

**DISPLAY** Kitta number, location, direction, Aana, Price, Months Rented, Months Returned, Monthly Rent and Extra Charge

**DISPLAY** “-“ times 130

**For** index i in kittas

**DO**

**FOR** each lines in the lands file

**DO**

**REPLACE** the ‘\n’ with blank space and split the words where there is ‘,’ and store it in list named line

**SET** the variable monthly\_price to 0

**SET** the variable extra\_charge to 0

**IF** the kitta number of line matches the kitta number in variable kittas with index i

**DO**

**MULTIPLY** monthly price from line with month list with index i and add it to monthly\_price

**IF** the months\_rented with index i is greater than months\_returned with index I or the months\_rented with index i equals months\_returned with index i

**DO**

**ADD** the monthly\_price to total\_price

**END DO**

**ELSE IF** the months\_rented with index I is lsss than months\_returned with index i

**DO**

**SUBTRACT** the months\_rented with index I from months\_returned with index I and assign the value to the variable remaining\_month

**MULTIPLY** the value of monthly price of line with 10% and then multiply the value with remaining\_month , then assign the value to extra\_charge

**Add** the monthly\_price and extra charge, then add it to the total\_price

**END DO**

**END IF**

**DISPLAY** all the details of line with appropriate spacing and

also the months having index i and monthly total

**DISPLAY** “-“ times 130

**END DO**

**END IF**

**END DO**

**END FOR**

**END DO**

**END FOR**

**DISPLAY** the value of total\_price\_display

**END DO**

## 2.4 Data Structures

Depending on the circumstance, data can be organized using data structures to facilitate more efficient access. Any programming language's foundational concepts, or data structures, are what a program is constructed upon. Compared to other programming languages, Python makes it easier to master the fundamentals of these data structures (GeeksforGeeks, 2023).

### 2.4.1 Lists

It is an ordered collection of objects. When creating a project in Python, it is one of the most important data structures. “Ordered collections” refers to a situation where every item in a list has an order that makes it easy to identify them. One fundamental feature of a list is that It doesn’t change the order of its element over the time (Taylor, 2023).

There are two types of lists:

1-D list:

It is the list which has only one dimensions and can store various data of similar type in a single list

2-D list:

It is the list which can use multiple dimensions and can store multiple type of same type of data. It is composed of two or more than two types of 1-D list

In my program I have used list to store multiple kitta and month given by user for multiple renting of lands. It makes it easier to print the bill later on and update the lands

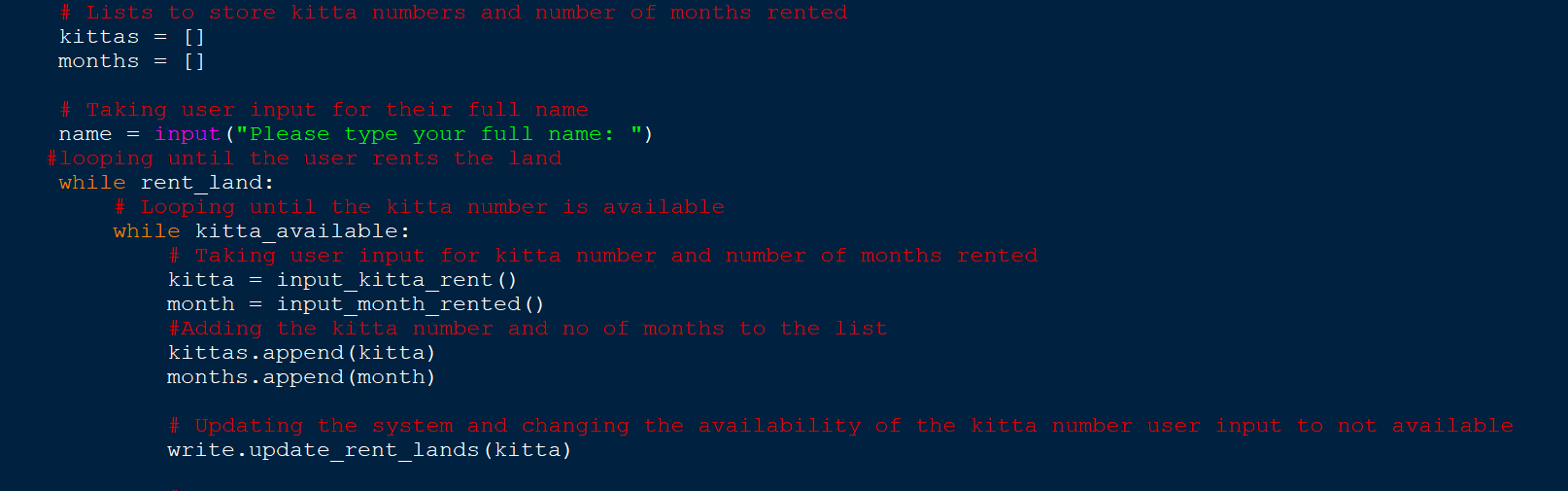


Figure 2: Use of list in my program

### 2.4.2 Dictonary

Python dictionaries are a type of data structure that are used to hold values in the key: value format. This distinguishes it from arrays, tuples, and lists as every key in a dictionary has a corresponding value. Dictionaries are ordered and cannot contain the same key twice. While keys in a dictionary cannot be duplicated and must be immutable, values in a dictionary can be of any data type (GeeksforGeeks, 2024).

**2.4.3 Tuple**

A tuple is an ordered collection of items and a built-in data structure in the Python programming language. Tuples have limited capabilities than list. A tuple can be distinguished from a list is by its mutability. Tuples are immutable, meaning they cannot be changed once they are generated whereas a list is mutable as one can change items of a list at any given time. The items of a list are enclosed in parenthesis and separated by commas to define the list (Taylor, 2023).

### 2.4.4 Sets

Python Sets are dynamic, unordered data collections that prohibit duplicate elements. In simple terms, sets are utilized for duplicate entry removal and membership checking (GeeksforGeeks, 2023). When an object's existence within a collection of objects matters more than its frequency of appearance or the arrangement of the objects, sets are utilized. Sets are changeable; they can be added, changed, replaced, or eliminated, in contrast to tuples (Taylor, 2023).

### 2.4.5 Int

In Python, entire Integers whether positive or negative are represented by the integer data type. Integers in python may be infinitely big without experiencing ant problem unlike other programming languages. They are a useful data type for a wide range of applications since they can be used for arithmetic and comparison operations such as addition subtraction multiplication and division as well as greater than, smaller than, equals to and so on. I used the int data type when I needed to get calculate the total price, get kitta number from user and so on.

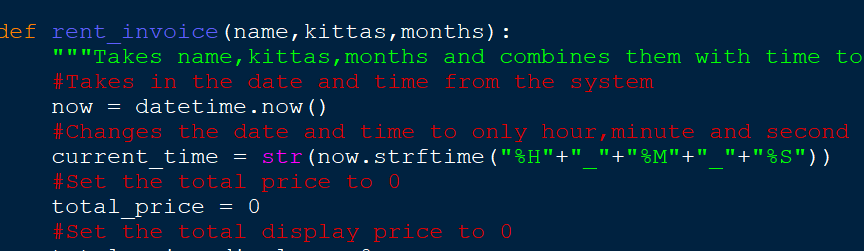


Figure 3: Use of int in my program

### 2.4.5 Float

Decimal numbers in Python are represented by the float data type. Due to the decimal point's ability to "float" in relation to the number's magnitude, floating point numbers are also referred to as floating-decimal or real numbers. There may be tiny errors in some computations due to rounding errors because floats, in contrast to integers, have a finite precision (Pierian Training, 2023).

### 2.4.6 String

Python represents text-based data using the string data type. Strings of characters are made up of letters, numerals, spaces, and other unique symbols, among others. either in two quotes or just one quote. They can hold names, addresses, and other text-based information. They offer several useful operations, such concatenation and slicing, that let developers structure them according to their requirements (Pierian Training, 2023).I have used string data type in places where I need to take name from the user.

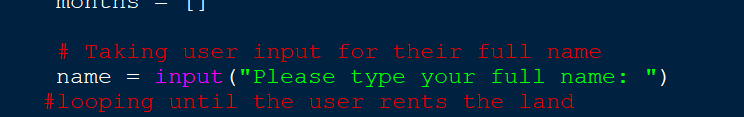


Figure 4: Use of string in my program

### 2.4.7 Boolean

In Python, logical values are represented by the Boolean data type and are often indicated as True or False. In order to make judgments in code using while loops, if-else statements, and other conditional expressions, Booleans are necessary. I have used Boolean data types in the program where I needed to run the loop till the user terminates the loop.

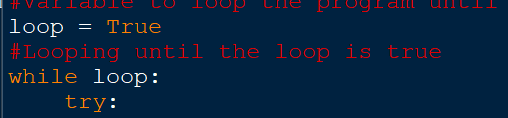


Figure 5: Use of Boolean in my program

# 3. Program

## 3.1 Implementation of program

This program is especially designed for the company named Techno Property Nepal as per the coursework. This program allows its clients to rent and return the lands in the system of the company easily. When the program is run, it lists out all the lands in the system showing their kitta number, location, direction, anna, monthly price and availability of the land and displays it to the user. Then the user is asked to chose if they want to rent a land or return a land or exit the program.

If the user chooses to rent the land, then the program displays all the available land in the system, then the program asks the user to provide their full name, after that the kitta number of the land they want to rent is asked and after the user input the program checks if the input land is available or not, in case the land is not available, A message is displayed saying that the land is not available or not in the system and the user is asked to input the kitta number again until the land provided is available, after taking the kitta number that is available, the program then asks the user to input the months they want to rent then the total price is calculated according to the months they want to rent the land. After that the user is asked if they want to rent another land, if the answer is yes then the whole process will again start from asking the user to input kitta number, if the answer is no then the unique invoice is generated which includes all the land rented with all their details with the total price of all the lands.

If the user chooses to return the land, the user is asked to input their name then the program asks the user to input the kitta number of the land they want to return. Then the program checks the land is in the system and isn’t available or not. If not in the system or available then a message is displayed to the user that the land is already returned or not in the system. If the land is in the system and not available then the user is asked how many months they rented and how many months after they are returning, if the month returned is greater than months rented additional 10% is charged for the extra months including the months till months returned, if the months returned is less than the months rented, only the price till the returned month is calculated then the user is asked if they want to rent another land, if the answer is yes then the process is again repeated from when the user is asked to input the kitta number if the answer is no then the invoice is generated and then printed on the screen.

## 3.2 Renting and Returning the land

When running the program, all the lands are displayed and then the user is asked to chose from one to three.

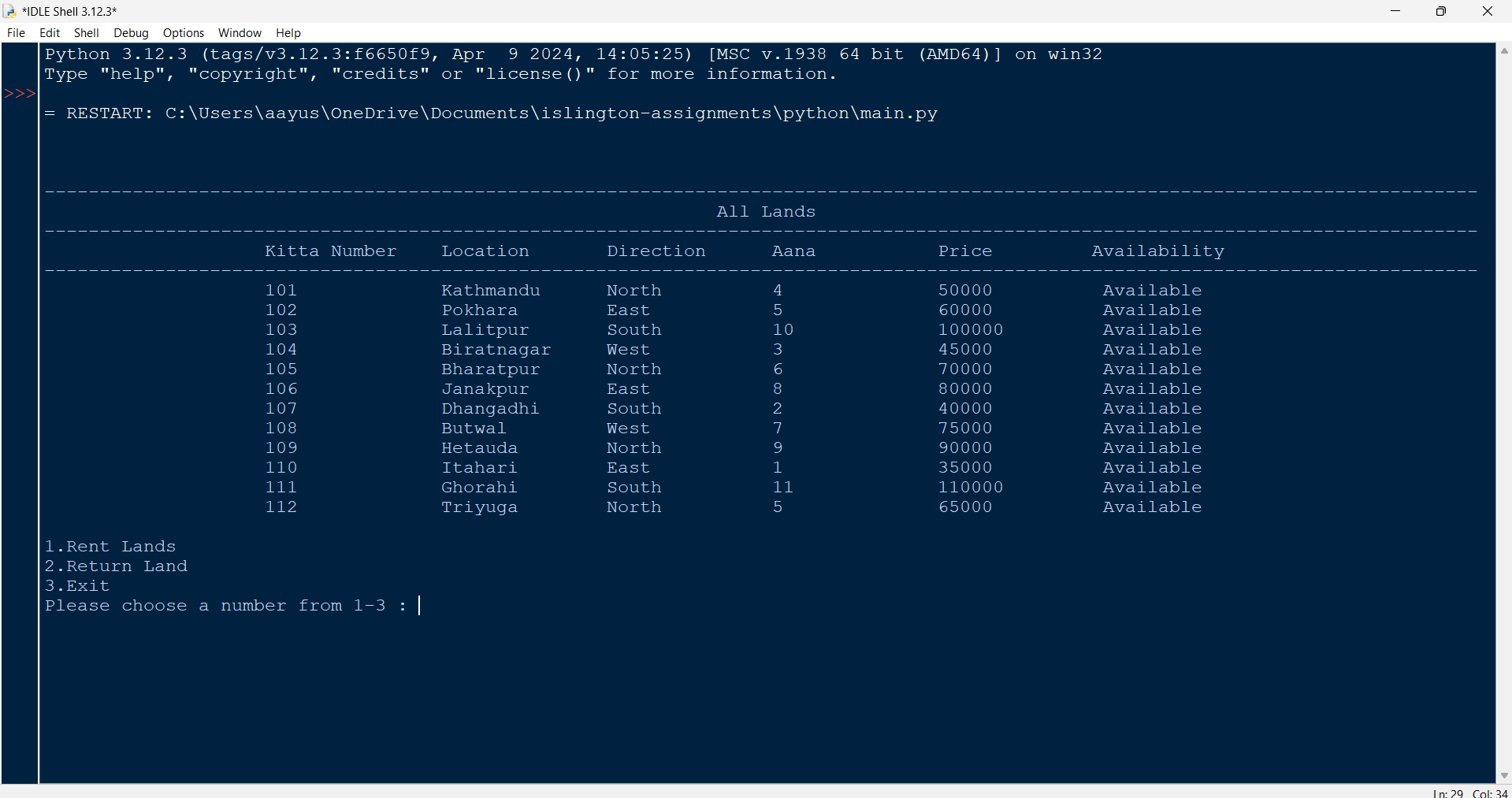


Figure 6: Renting and returning the land – 1

When giving the value of 1, all the available lands are displayed. And user is asked to give their full name.

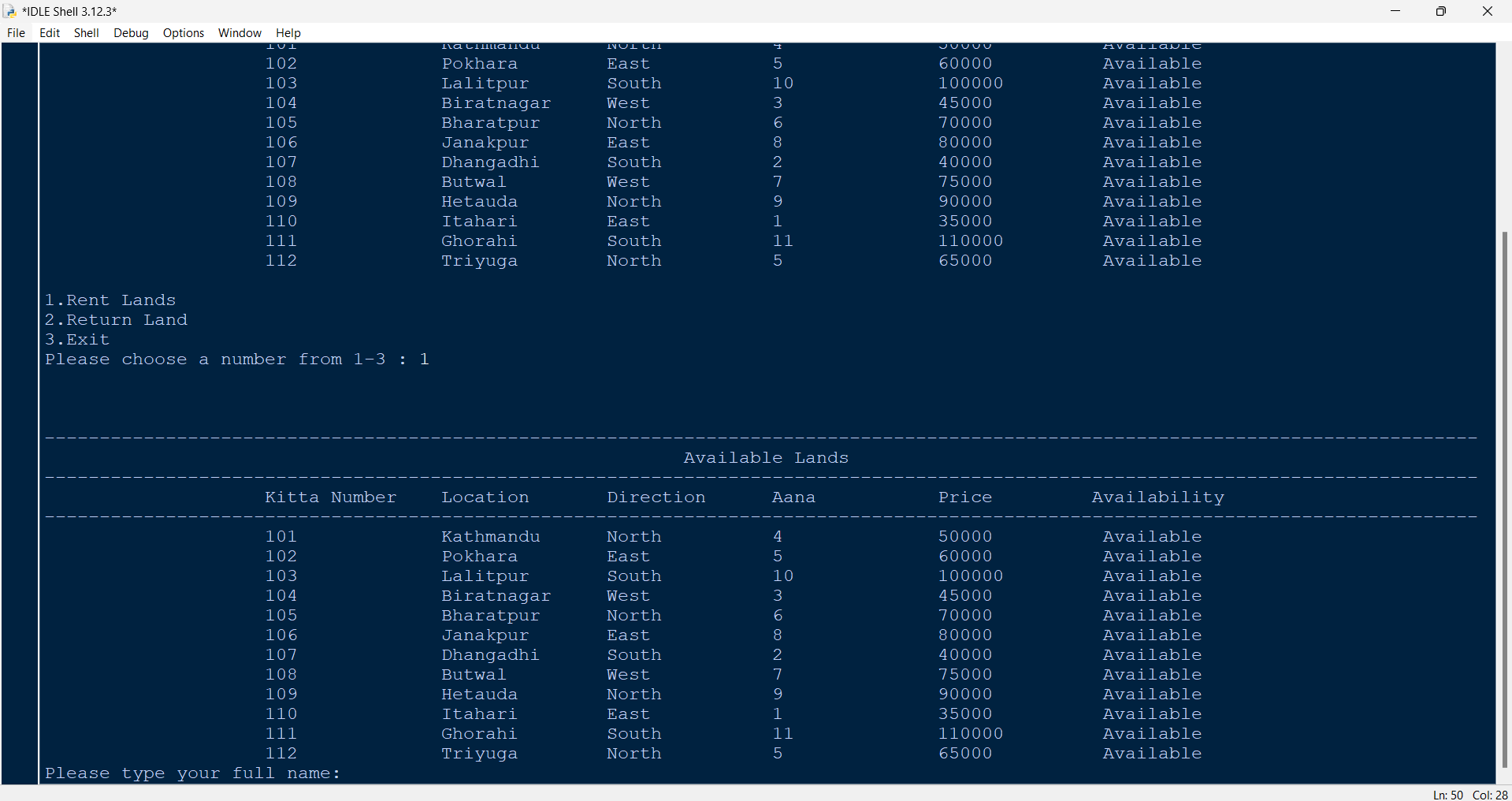


Figure 7: Renting and returning the land – 2

After giving their name, the user is asked to give the kitta number of the land they want to rent.

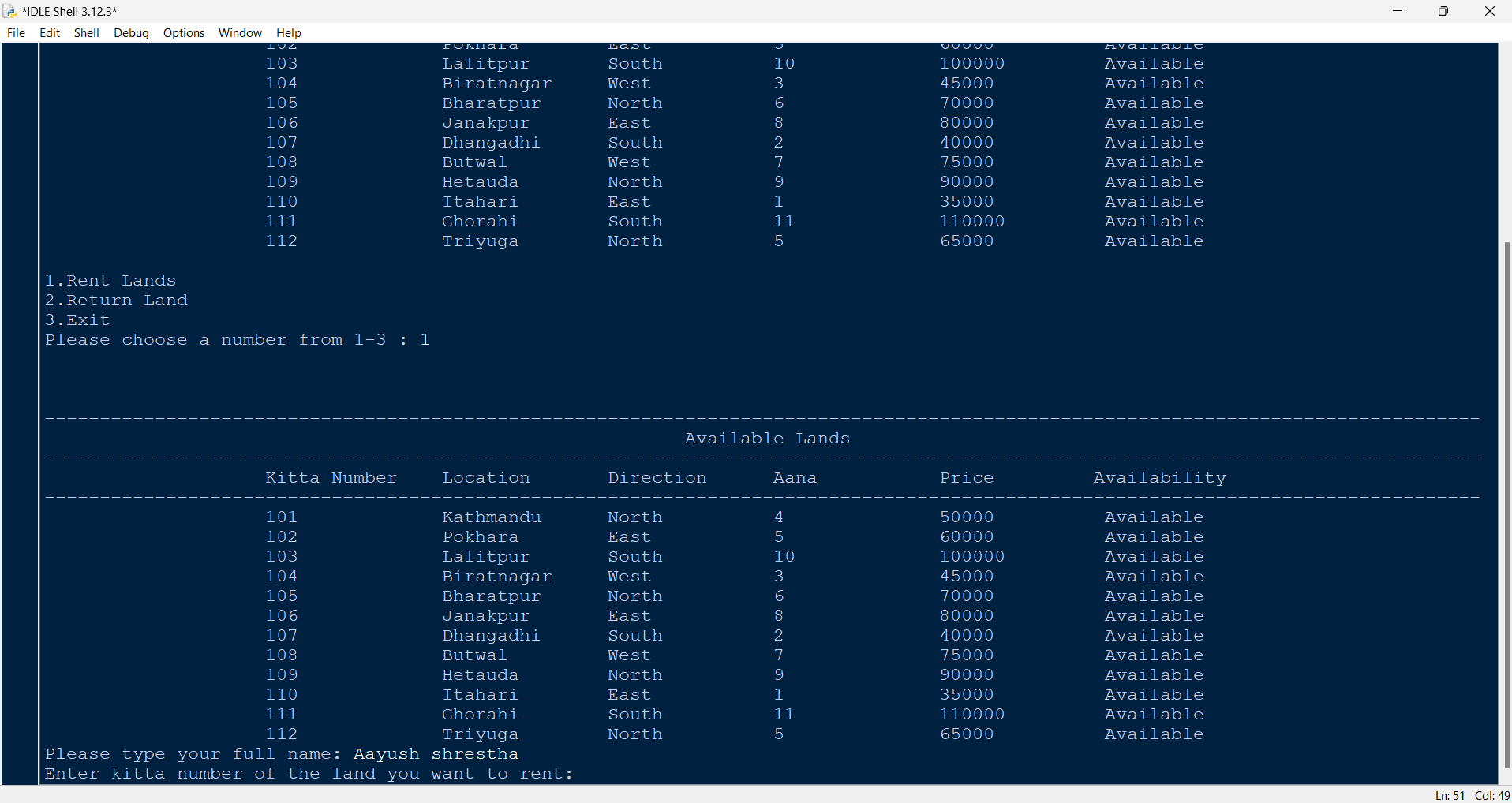


Figure 8: Renting and returning the land – 3

After giving kitta number the user is asked to give the number of months they want to rent for.

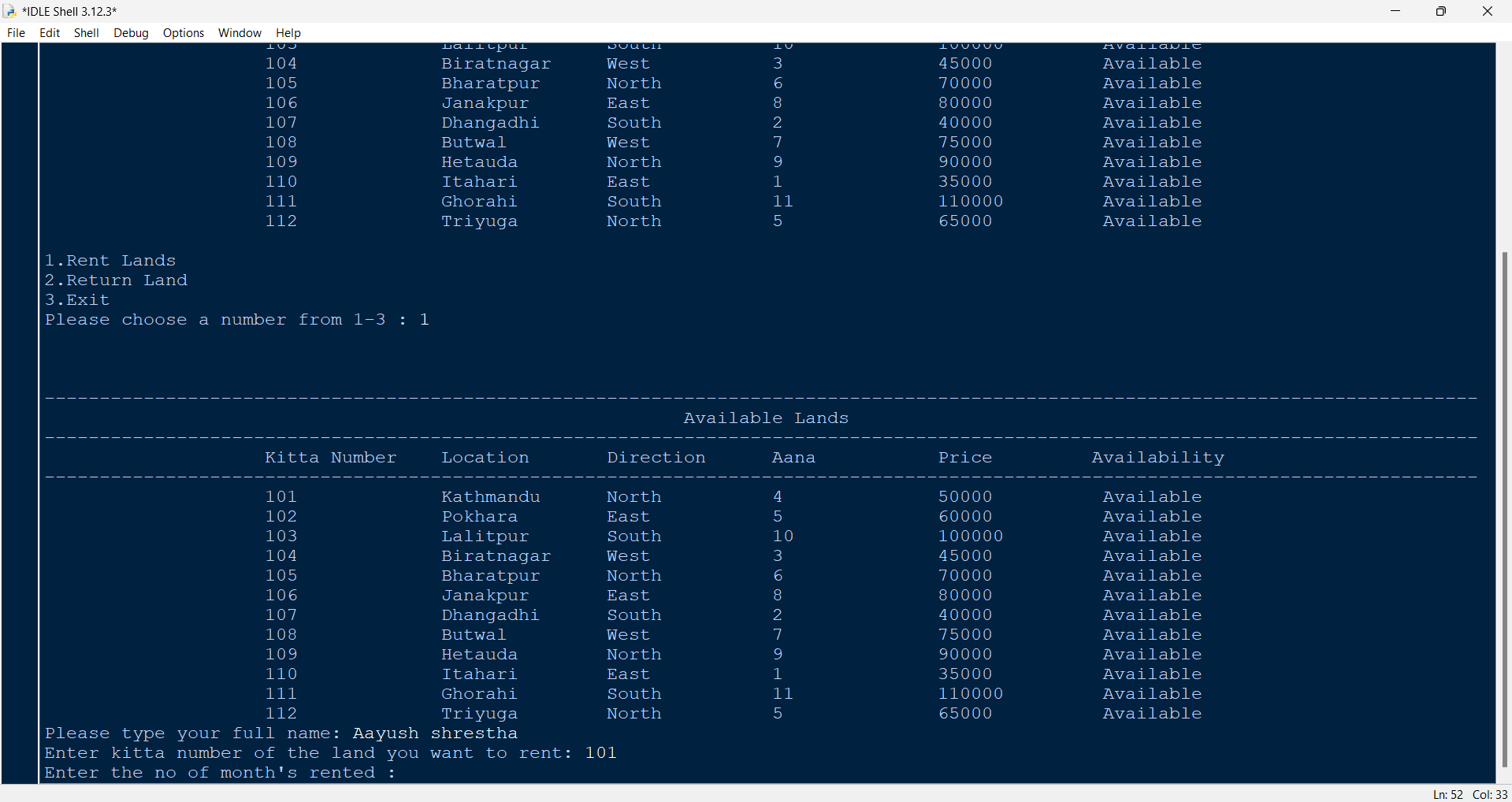


Figure 9: Renting and returning the land – 4

After giving the month, the user is asked if they want to rent another land.

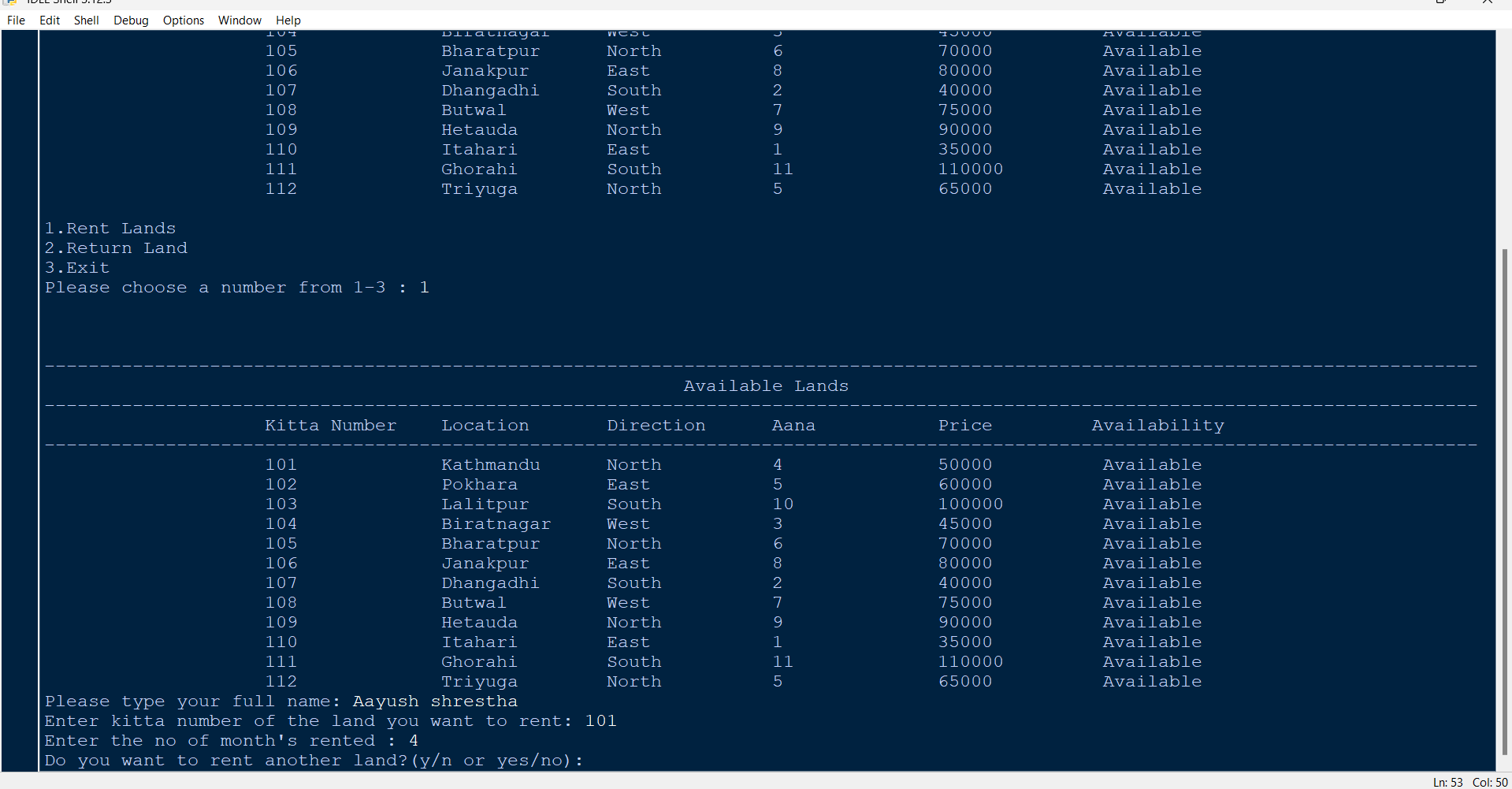


Figure 10: Renting and returning the land – 5

If the answer is no, the bill will be generated and displayed on screen.

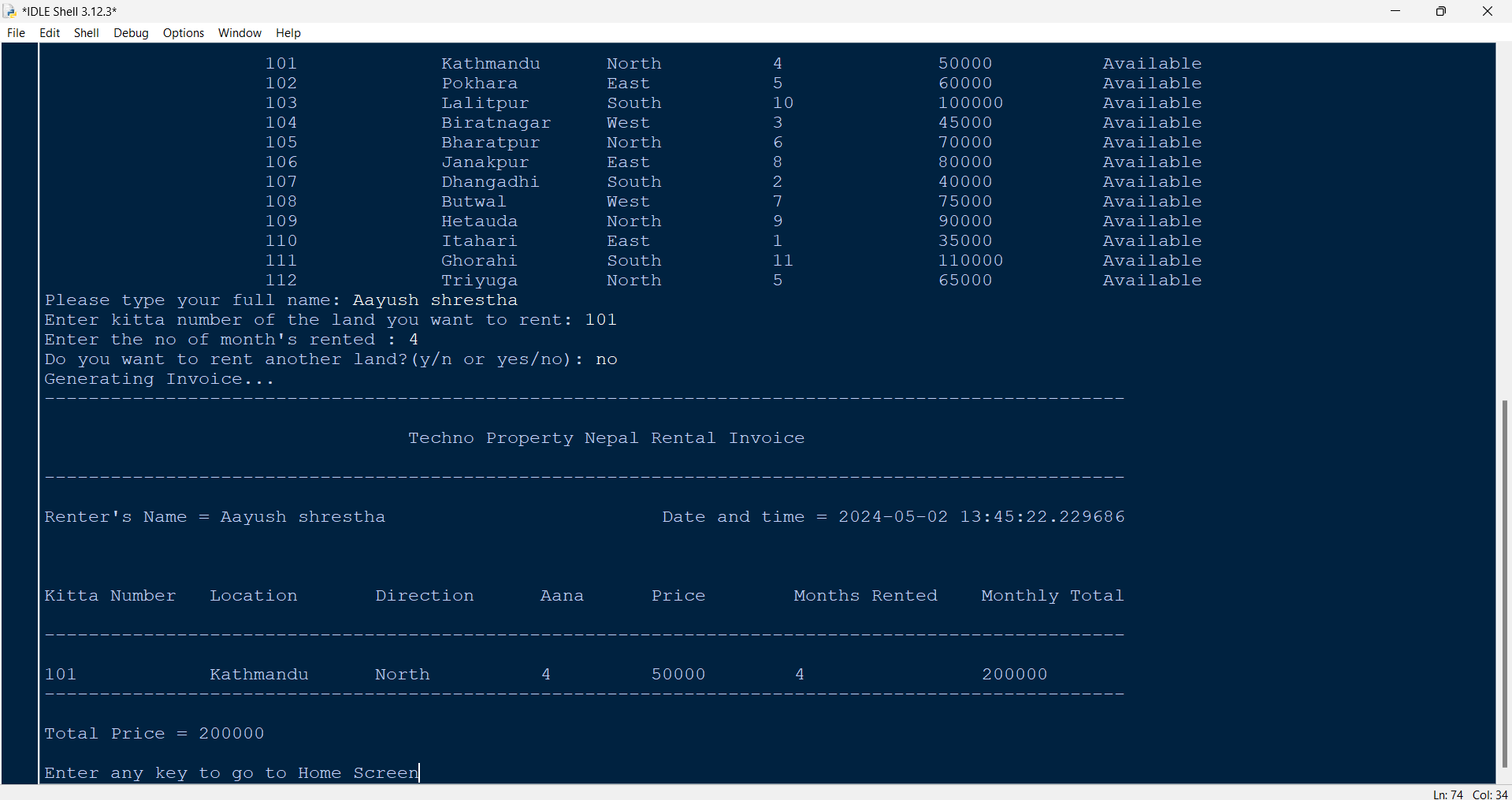


Figure 11: Renting and returning the land – 6

After pressing enter, we are redirected to main screen and the land that we rented is then changed to not available.

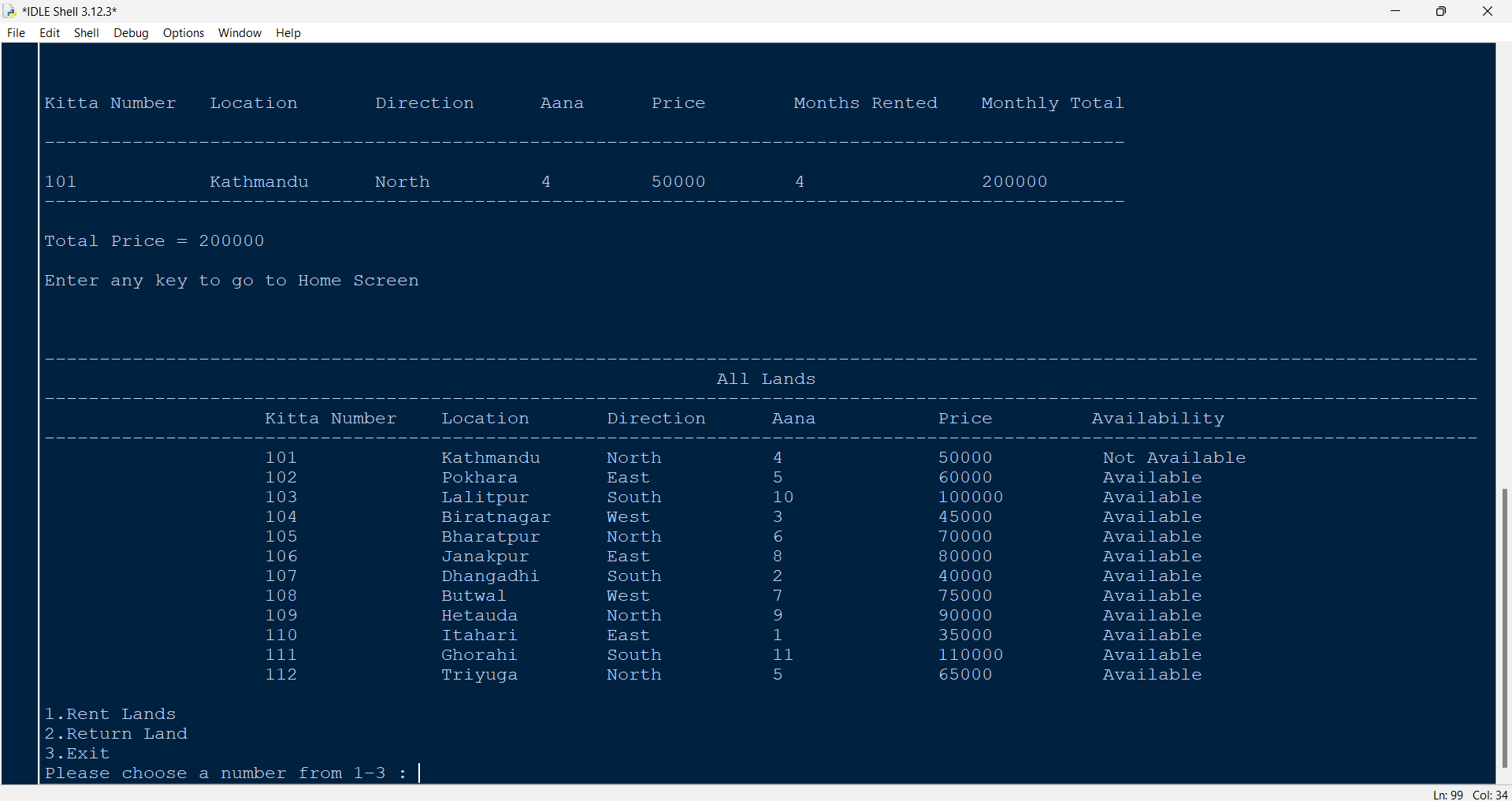


Figure 12 :Renting and returning the land – 7

When the user inputs 2, the user is asked to enter their full name.

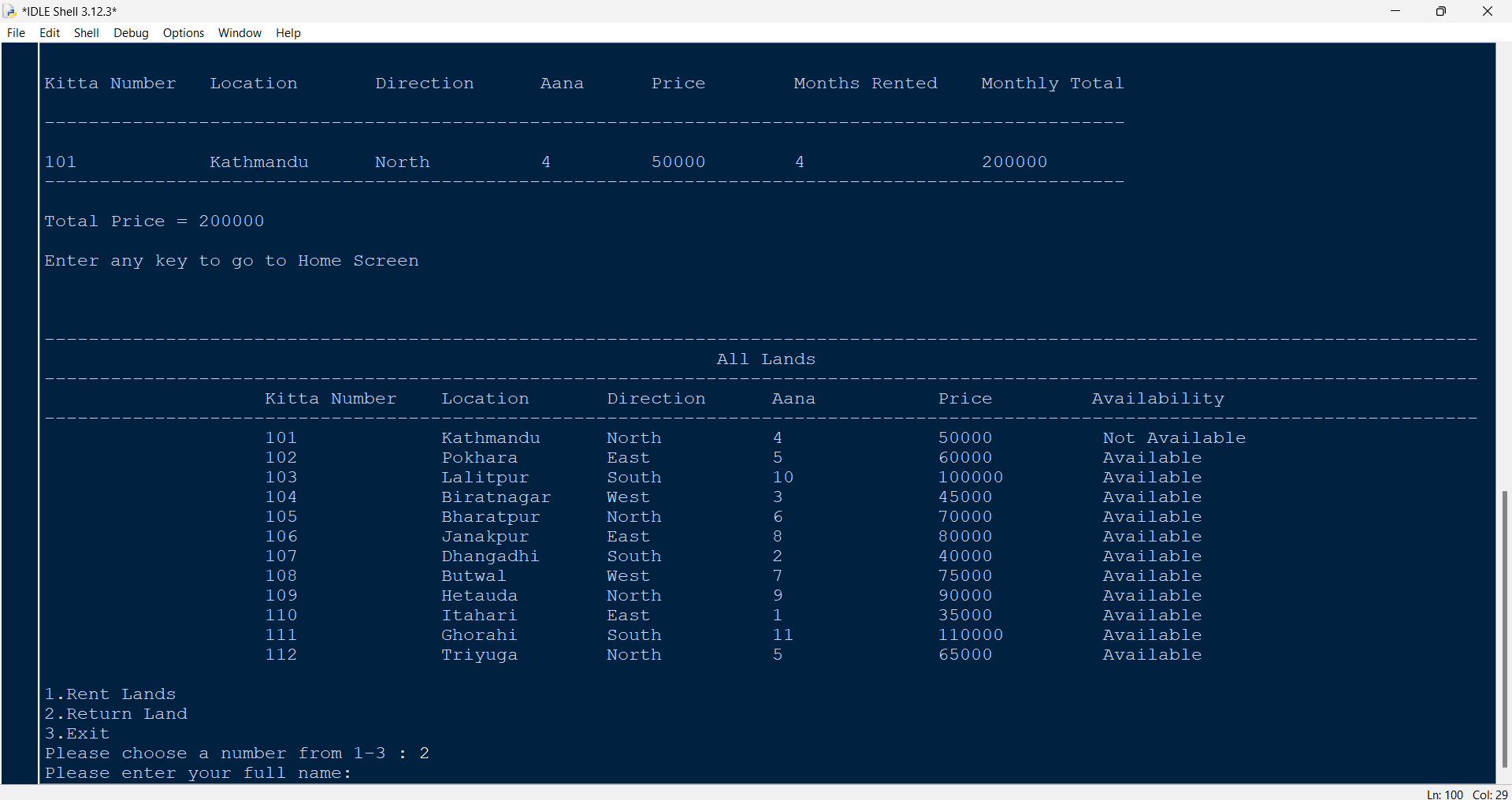


Figure 13: Renting and returning the land – 8

After giving the name, the user is asked to enter the kitta number of the land they want to return.

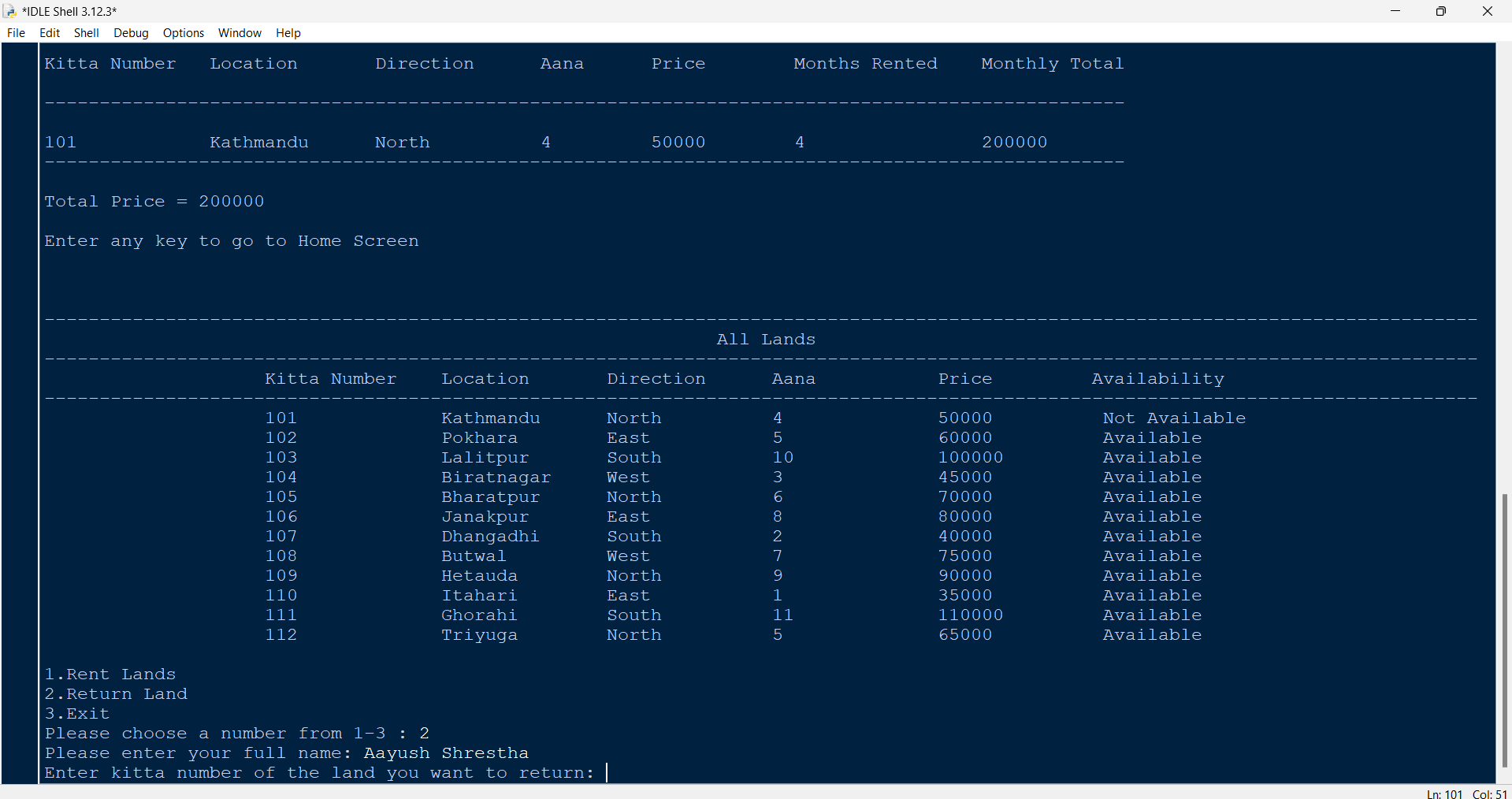


Figure 14: Renting and returning the land – 9

After giving the kitta number, the user is asked to enter the no of month rented.

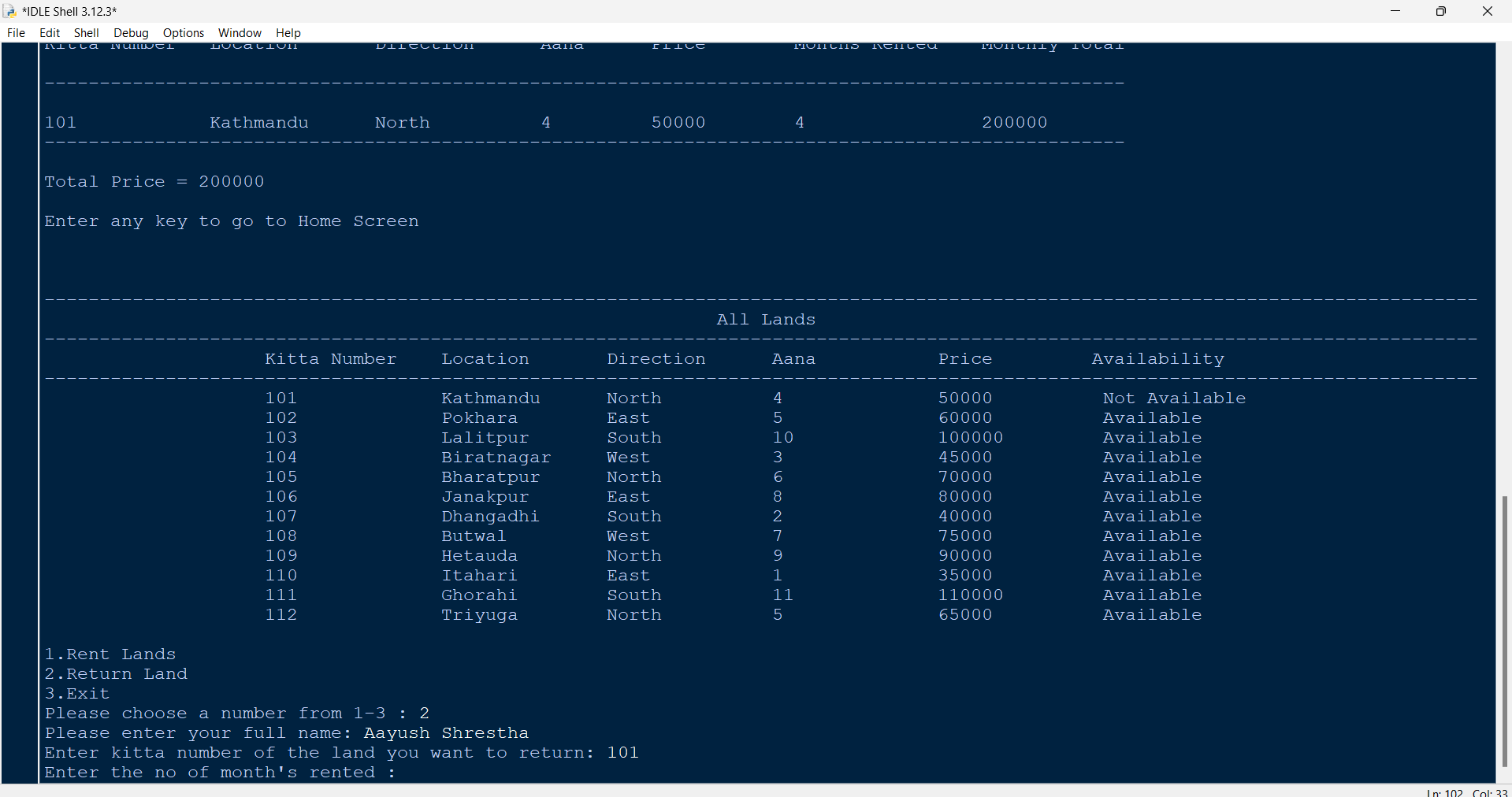


Figure 15: Renting and returning the land – 10

After that, they are prompt to enter the number of month after they are returning.

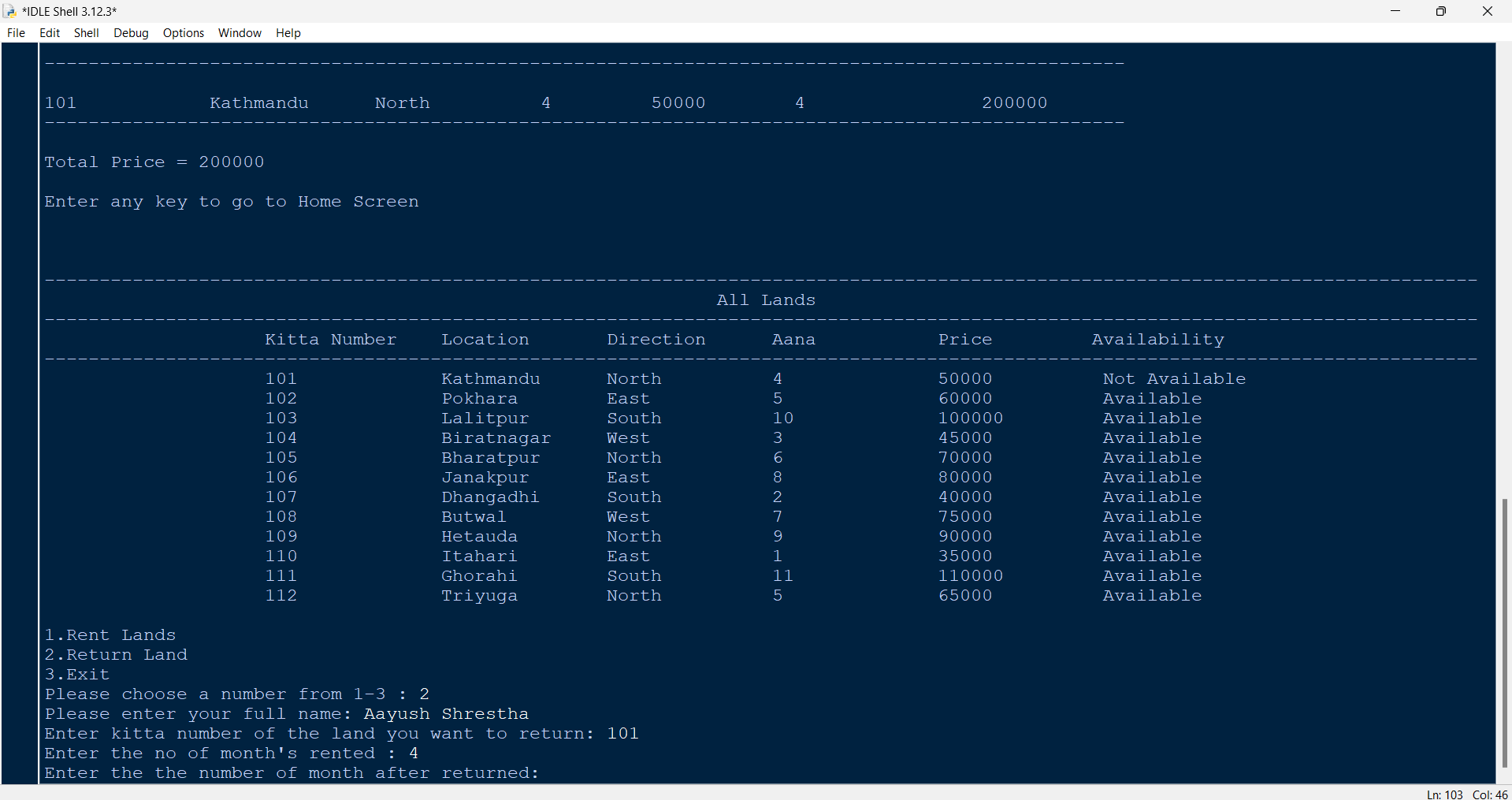


Figure 16: Renting and returning the land – 11

Then the user is asked if they want to return another land, if the answer is no then the bill is generated.

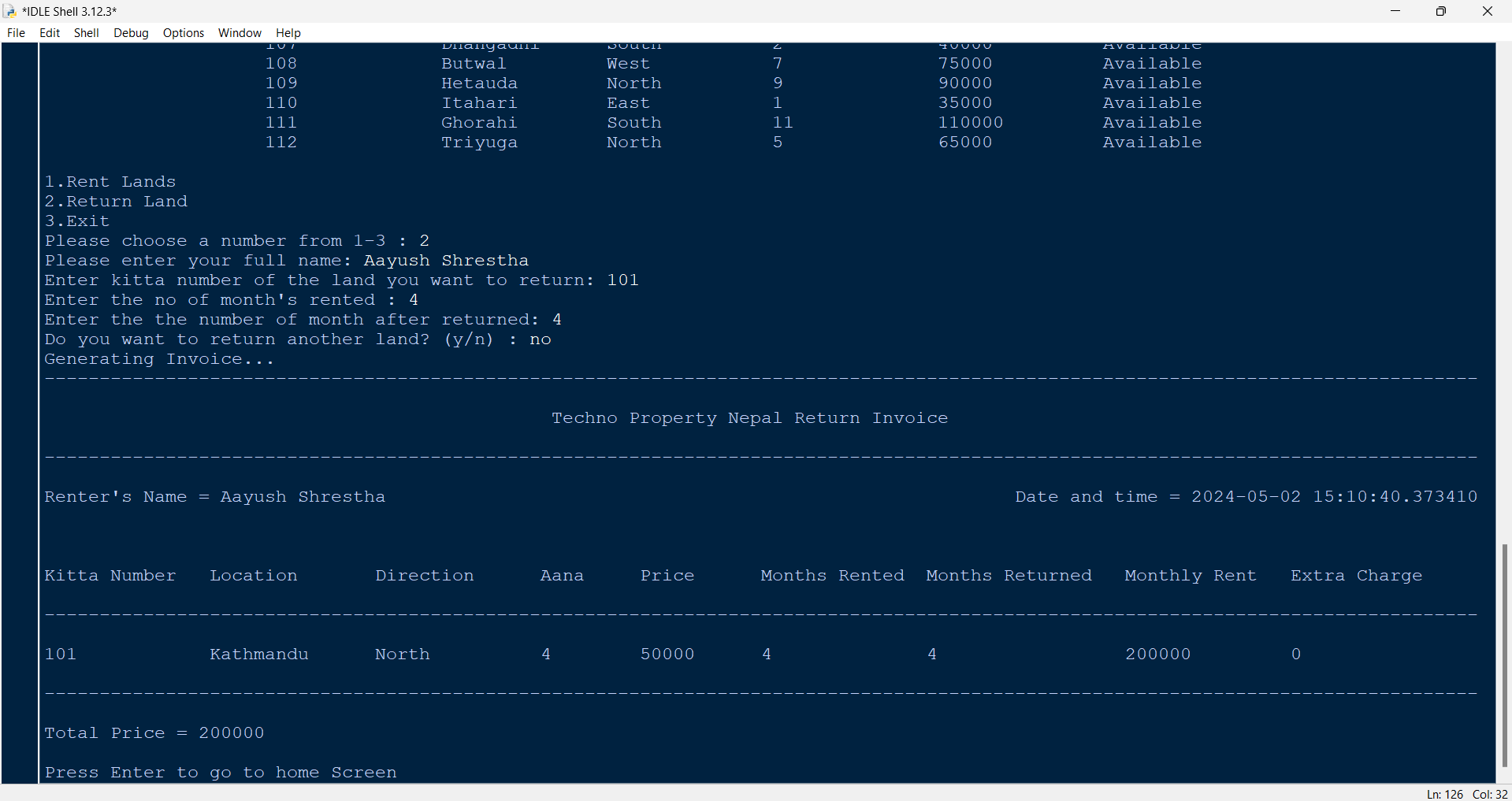


Figure 17: Renting and returning the land – 12

After that, the user is redirected to home screen with the updated land details.

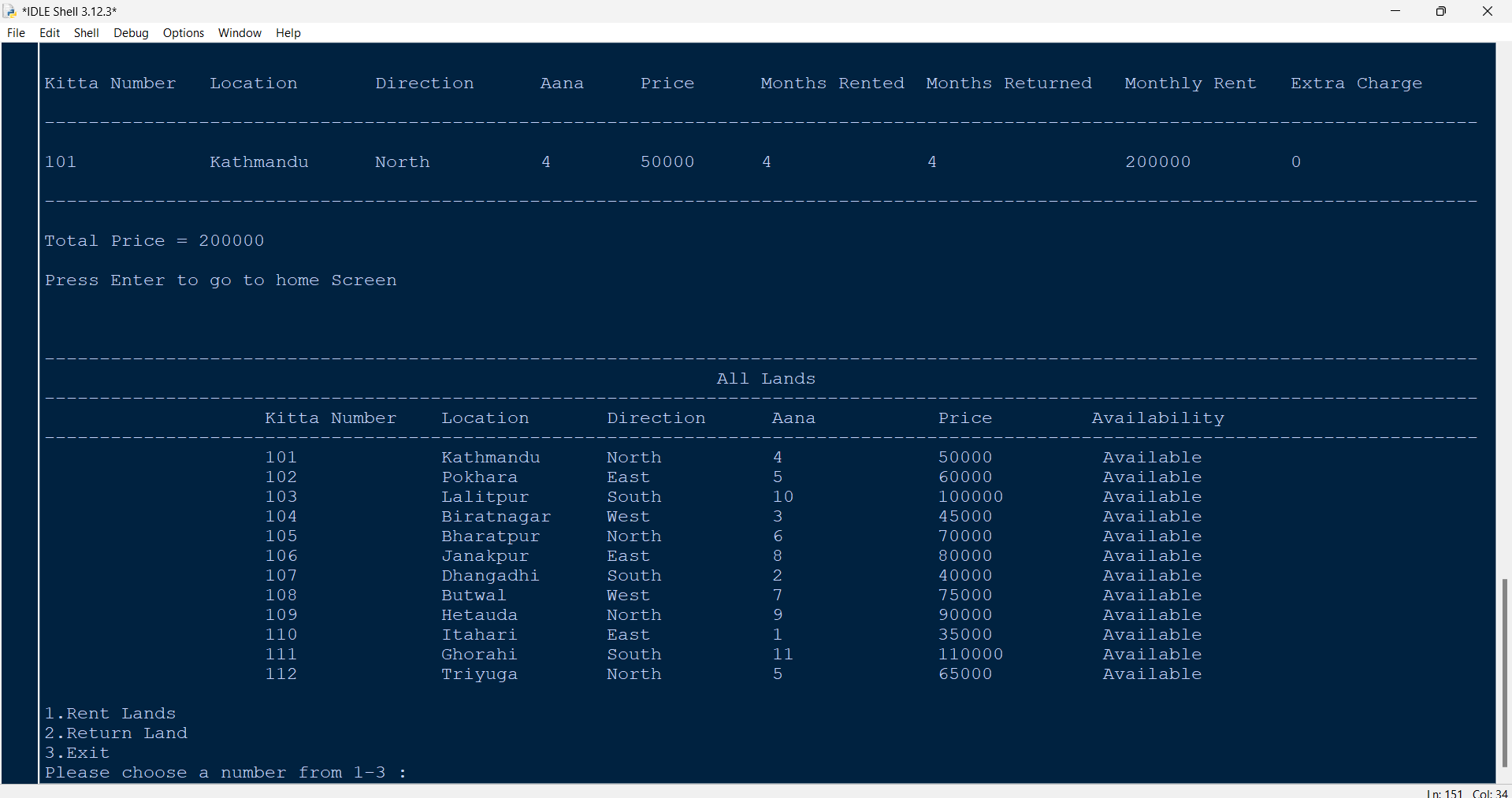


Figure 18: Renting and returning land – 13

Now the renting and returning file are generated in the program’s folder with unique file names.

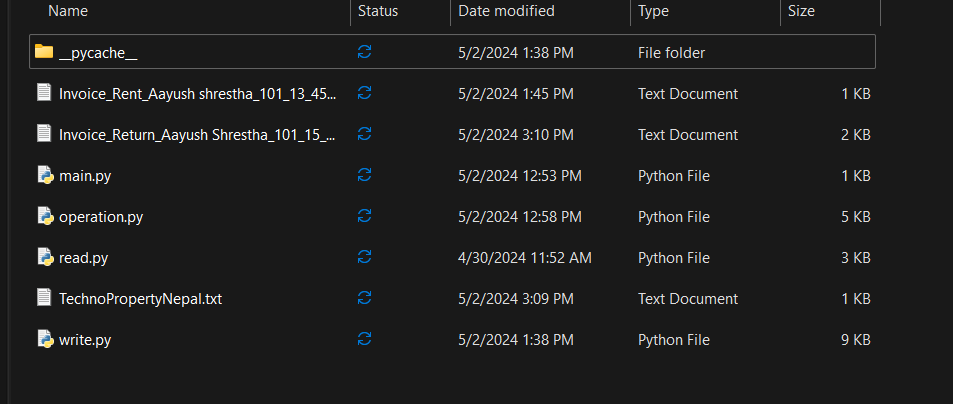


Figure 19: Renting and returning land – 14

The invoice of the rent are as follows:

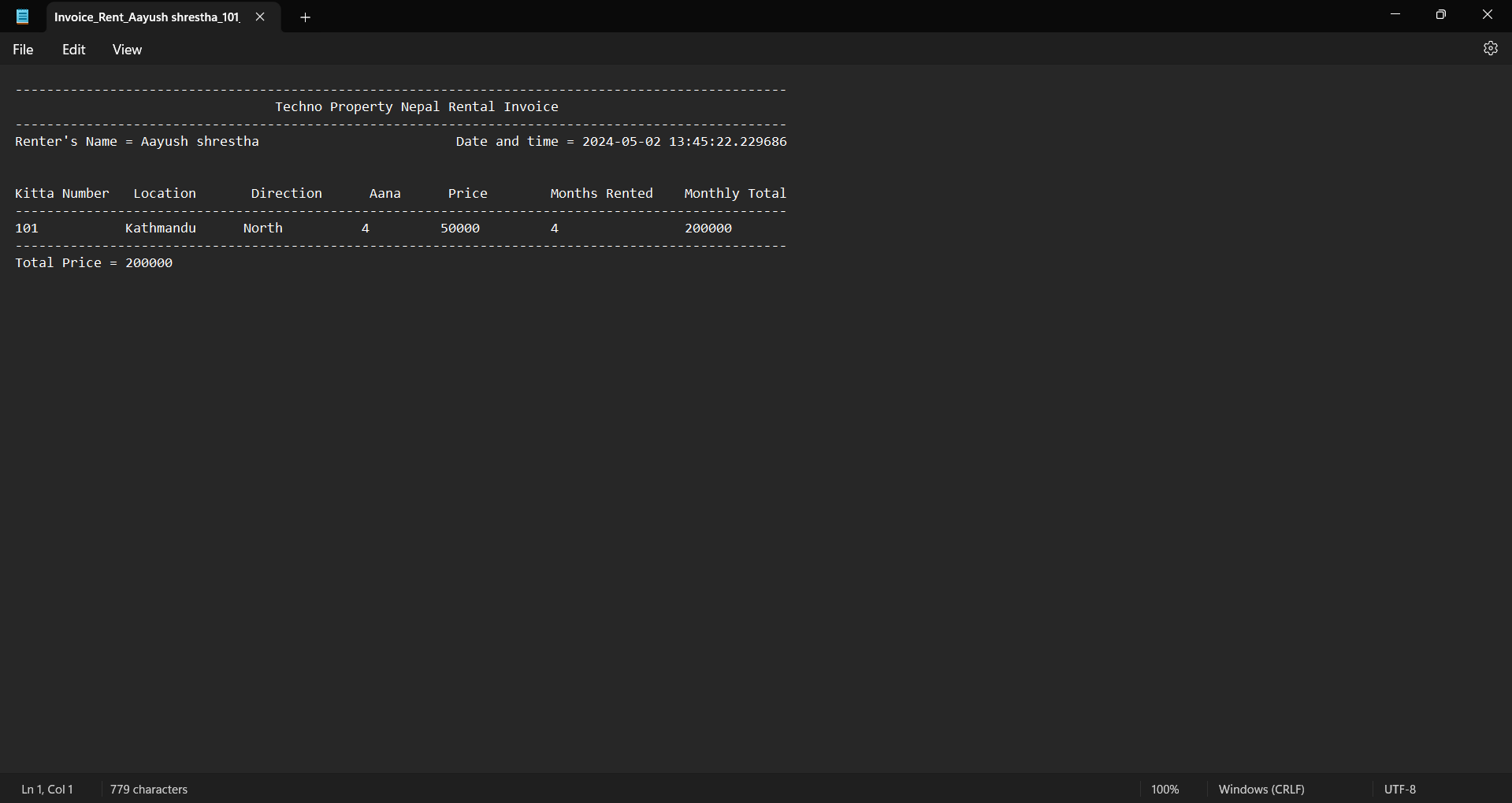


Figure 20: Renting and returning land – 15

The invoice of the return are as follows:

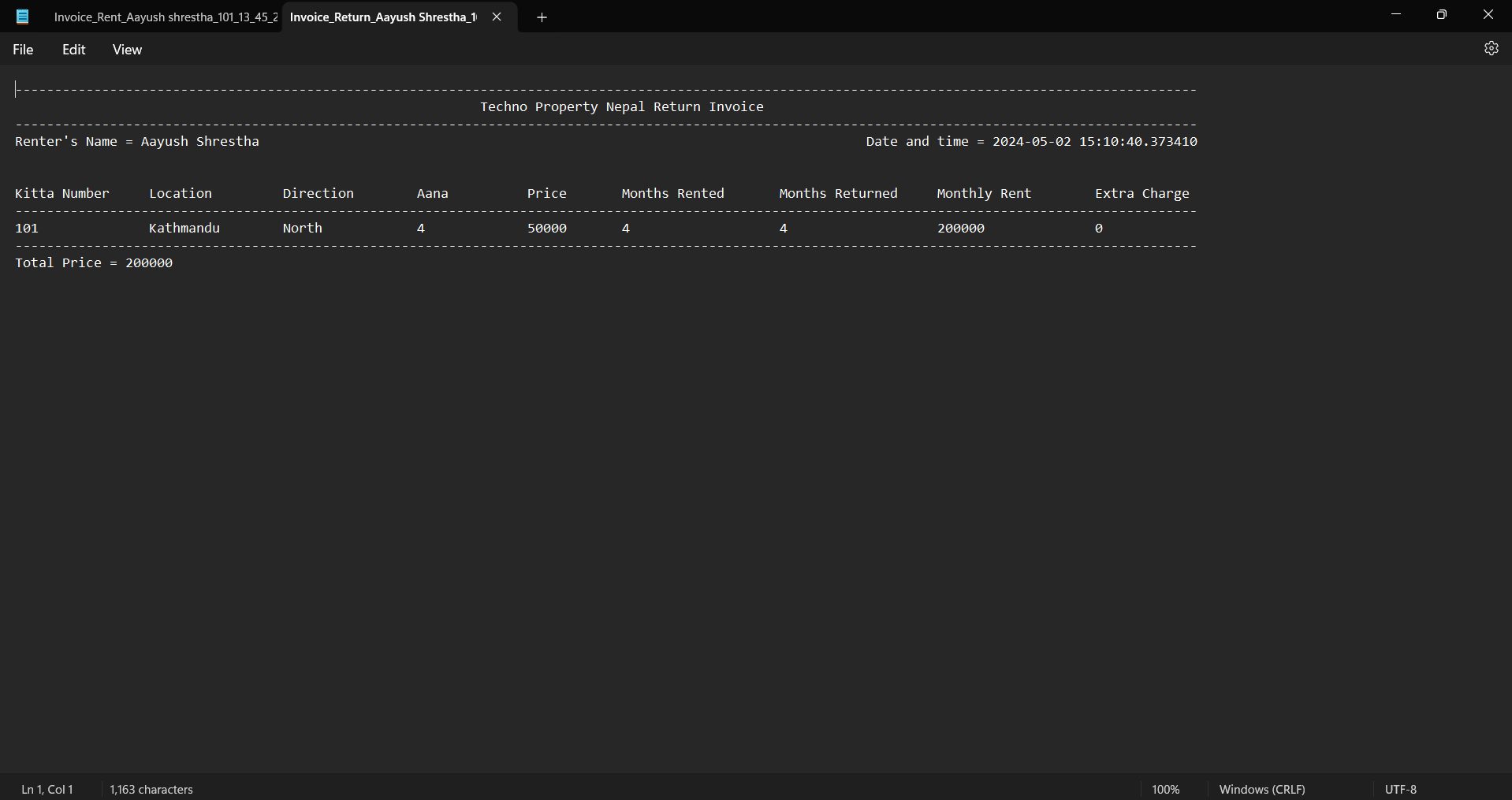


Figure 21: Renting and returning land - 16

To terminate the program, in the main screen, typing the 3 will terminate the program.

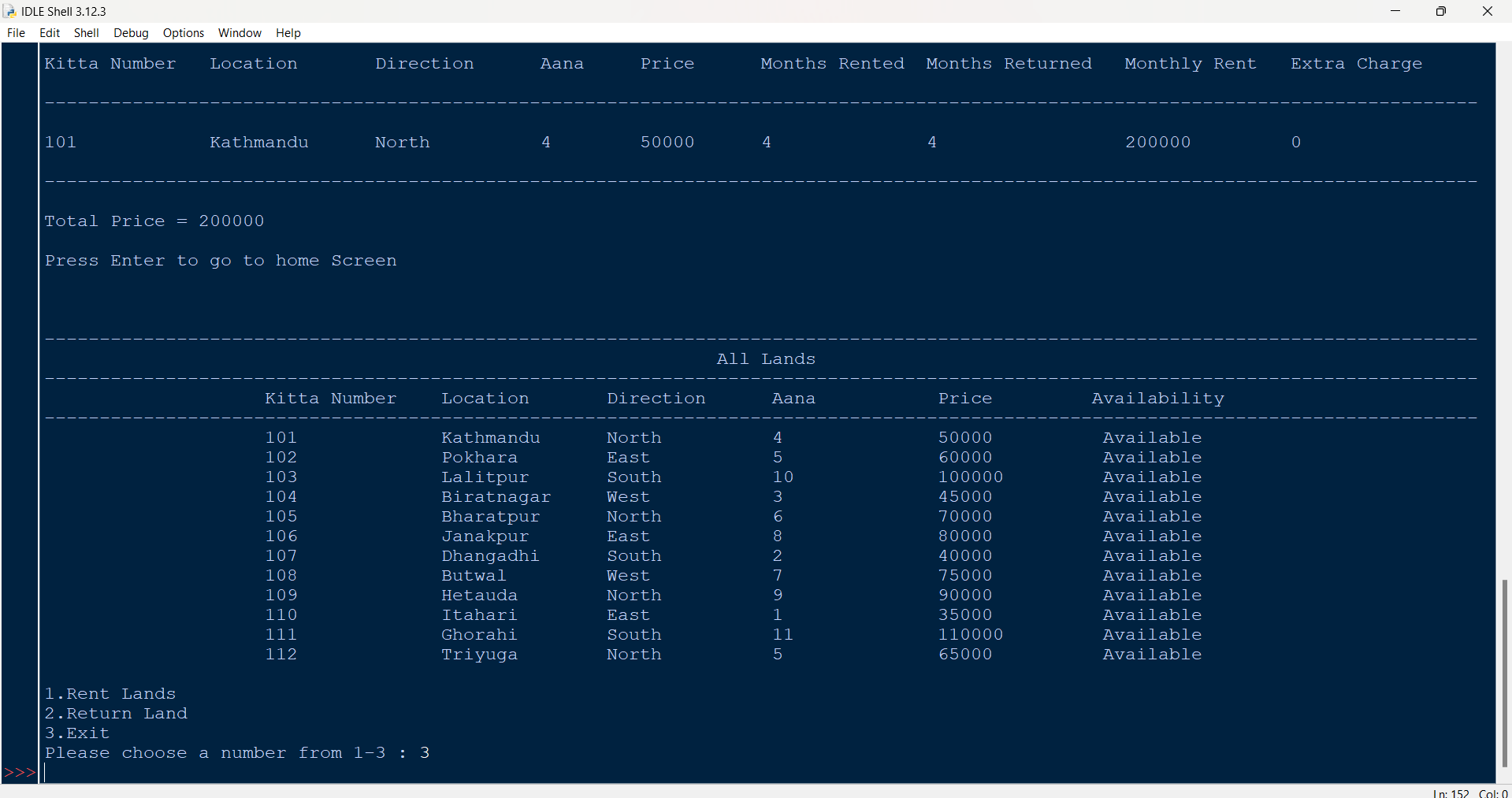


Figure 22: Renting and returning the land – 19

# 4. Testing

## 4.1 Test – 1: To show implementation of try, except

|  |  |
| --- | --- |
| Test no: | 1 |
| Objective | To show implementation of try, except |
| Action | Run the program  In the home screen, 1 is given as input,  Then, in the name, Aayush Shrestha is given as input,  After that, in the kitta number a is given as input |
| Expected Result | It will show Invalid input and ask user to re enter the kitta number. |
| Actual Result | It showed invalid input and asked the user to re-enter a kitta number. |
| Conclusion | The test is successful |

Table 1: Test -1: To show Implementation of try, except

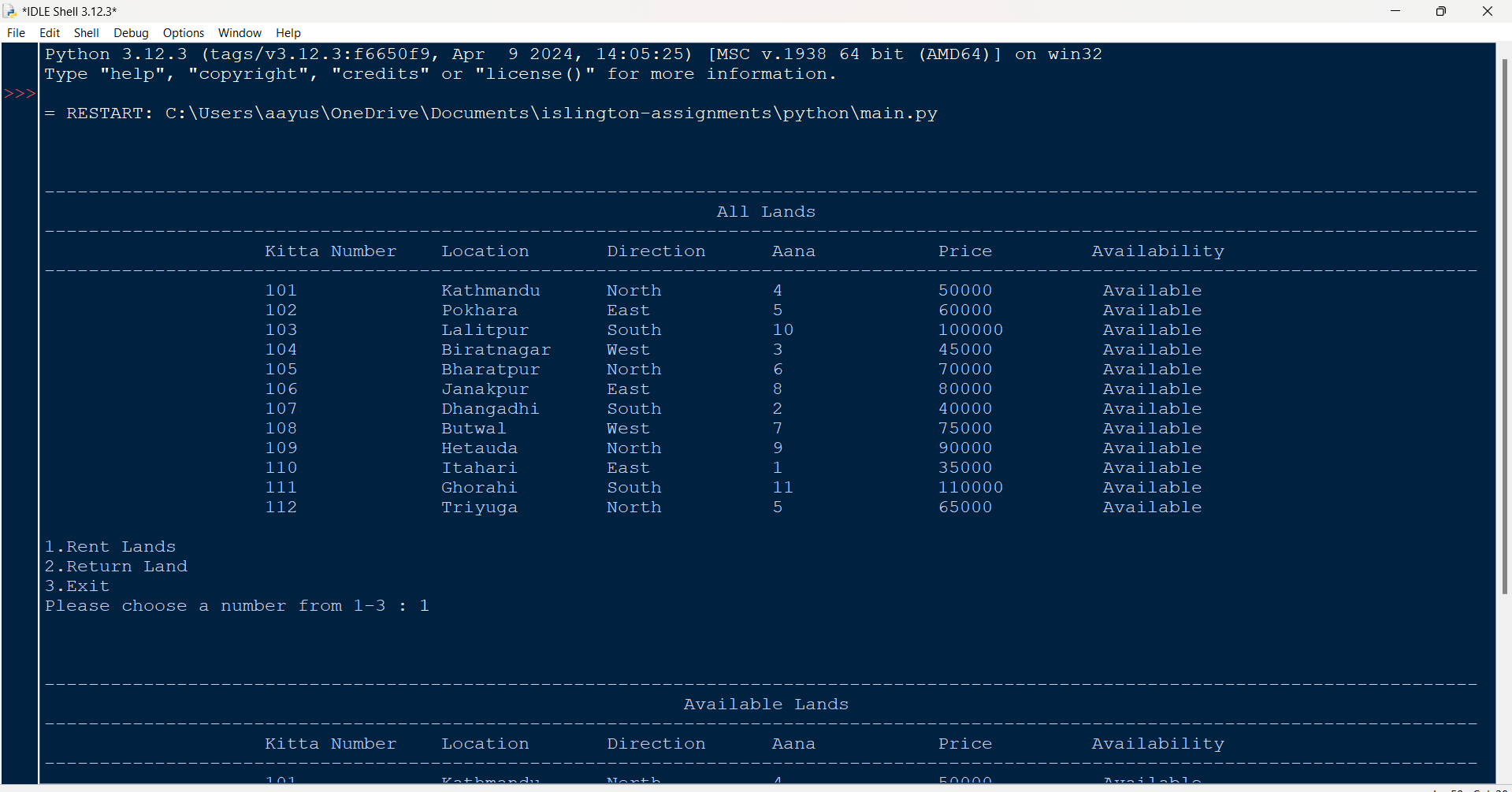


Figure 23: Test -1 – Choosing 1 in main menu

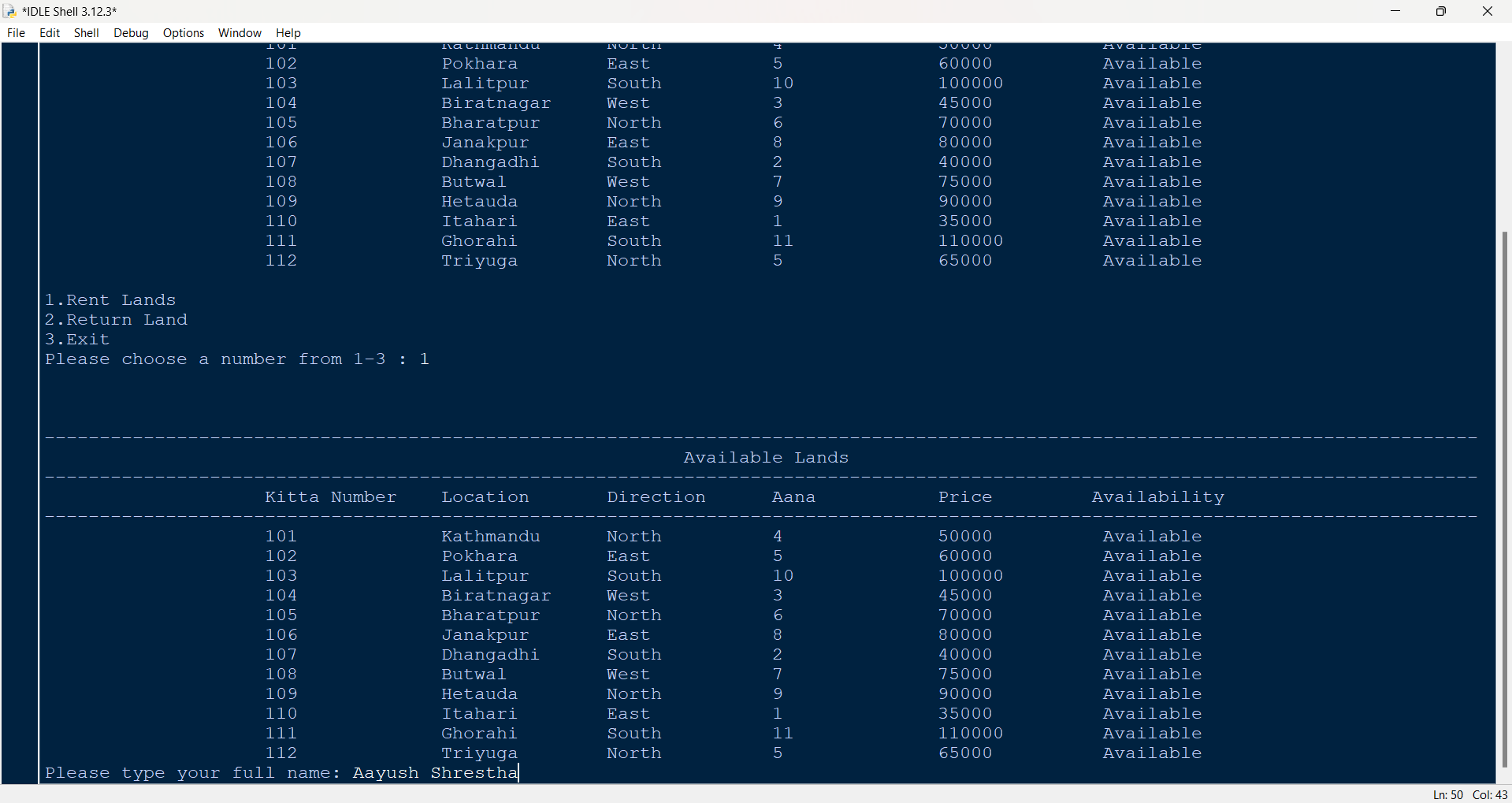


Figure 24: Test -1 - Giving name as input



Figure 25: Test -1 – Giving a as input in input kitta

## 4.2 Test – 2: To select rent and return of lands

|  |  |
| --- | --- |
| Test no: | 2 |
| Objective | To select rent and return of lands |
| Action | Run the program  In the home screen, 1 is given as input,  Then, in the name, Raman Shrestha is given as input,  After that, in the kitta number 201 is given as input  Again, in kitta number 101 is given as input,  In the rented months, -10 is given as input |
| Expected Result | It will show an error message that the land is not available in the system when 201 is given as input and then when rented month is given as -10 it will display an error message saying Invalid Input, months rented must be between 1-12. |
| Actual Result | It displayed the error messages |
| Conclusion | The test is successful |

Table 2: Test - 2: To select rent and return of lands

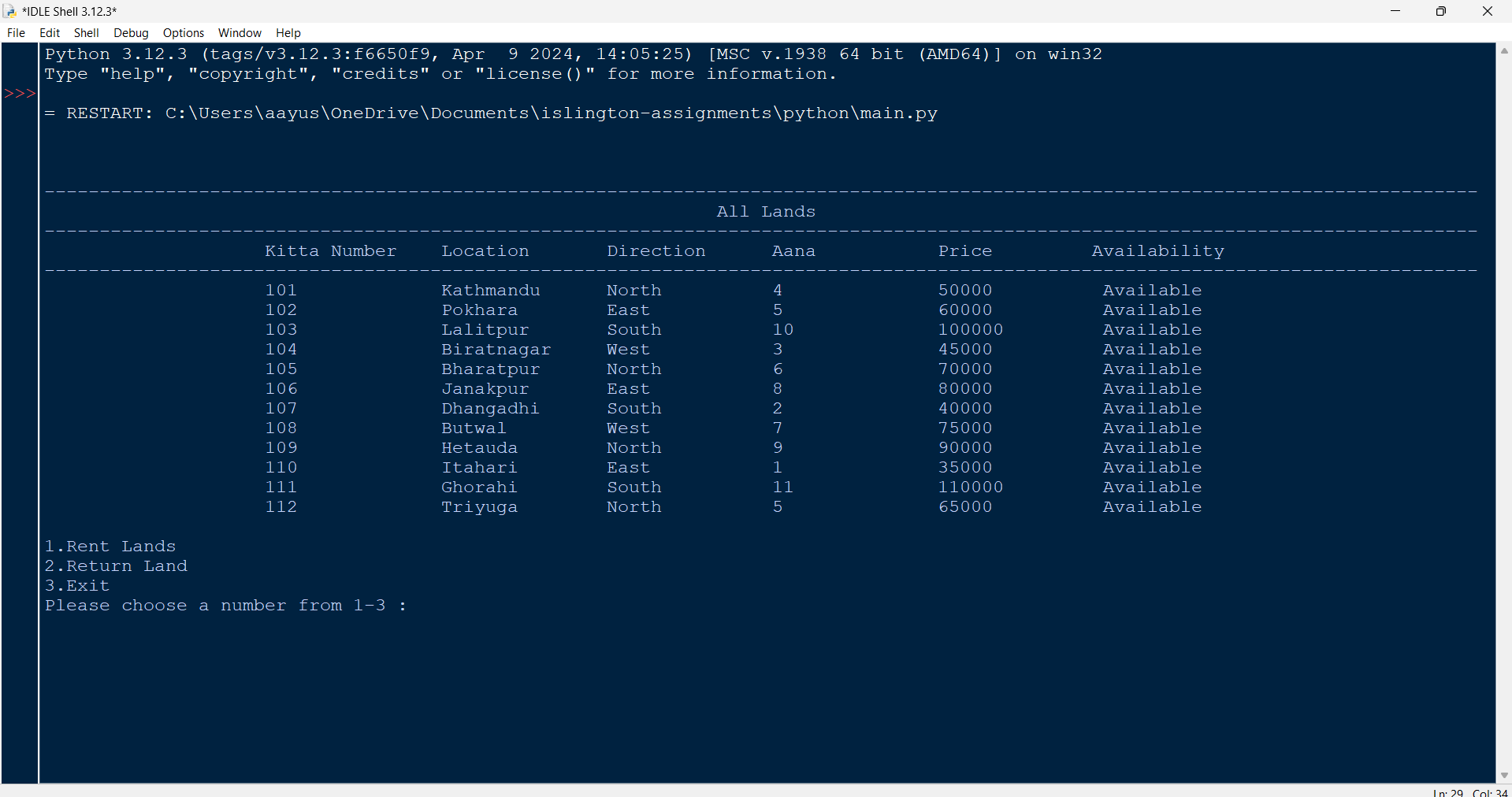


Figure 26: Test -2: To select rent and return of lands - 1

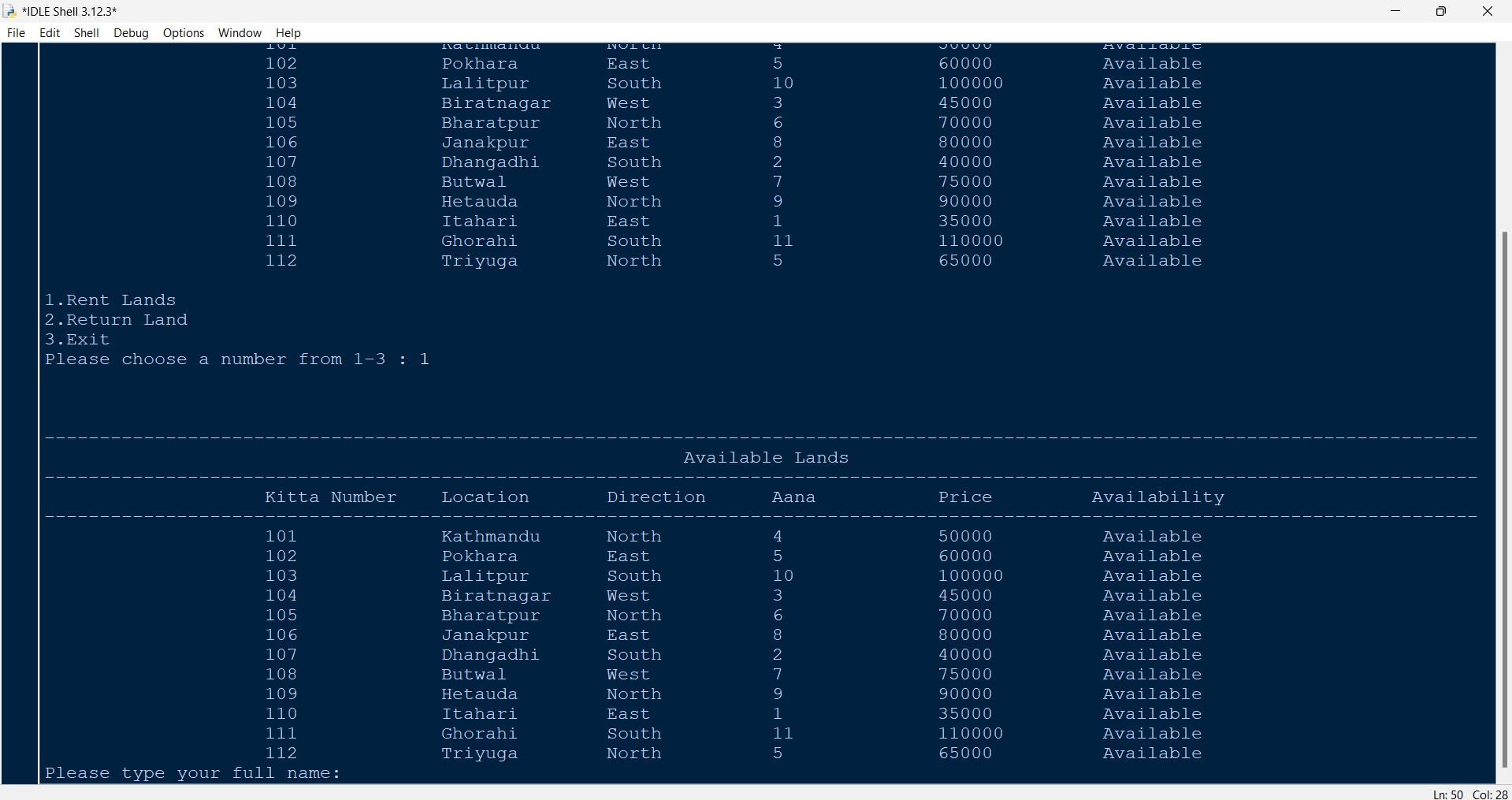


Figure 27: Test -2: To select rent and return of lands - 2

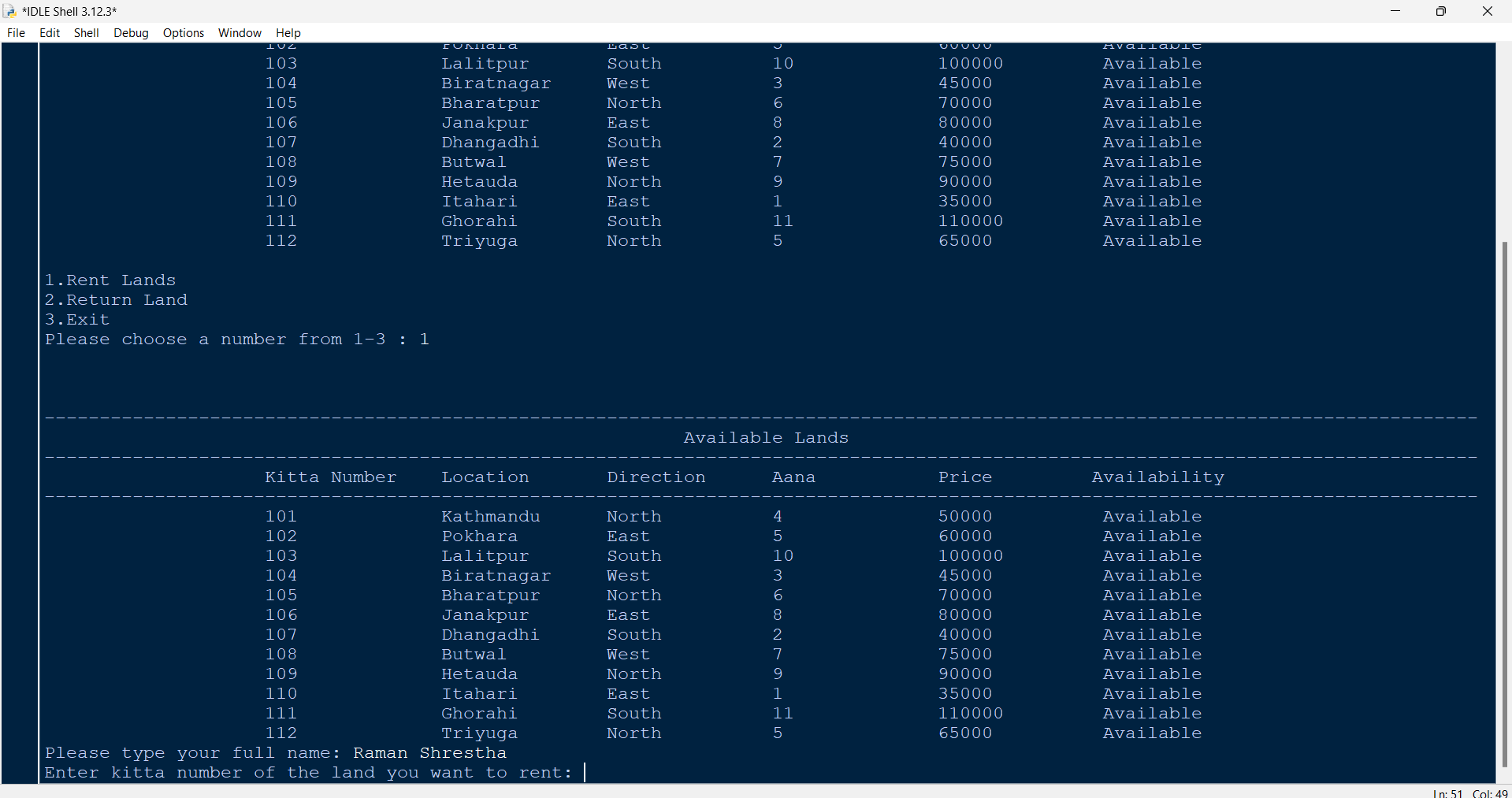


Figure 28: Test -2: To select rent and return of lands - 3

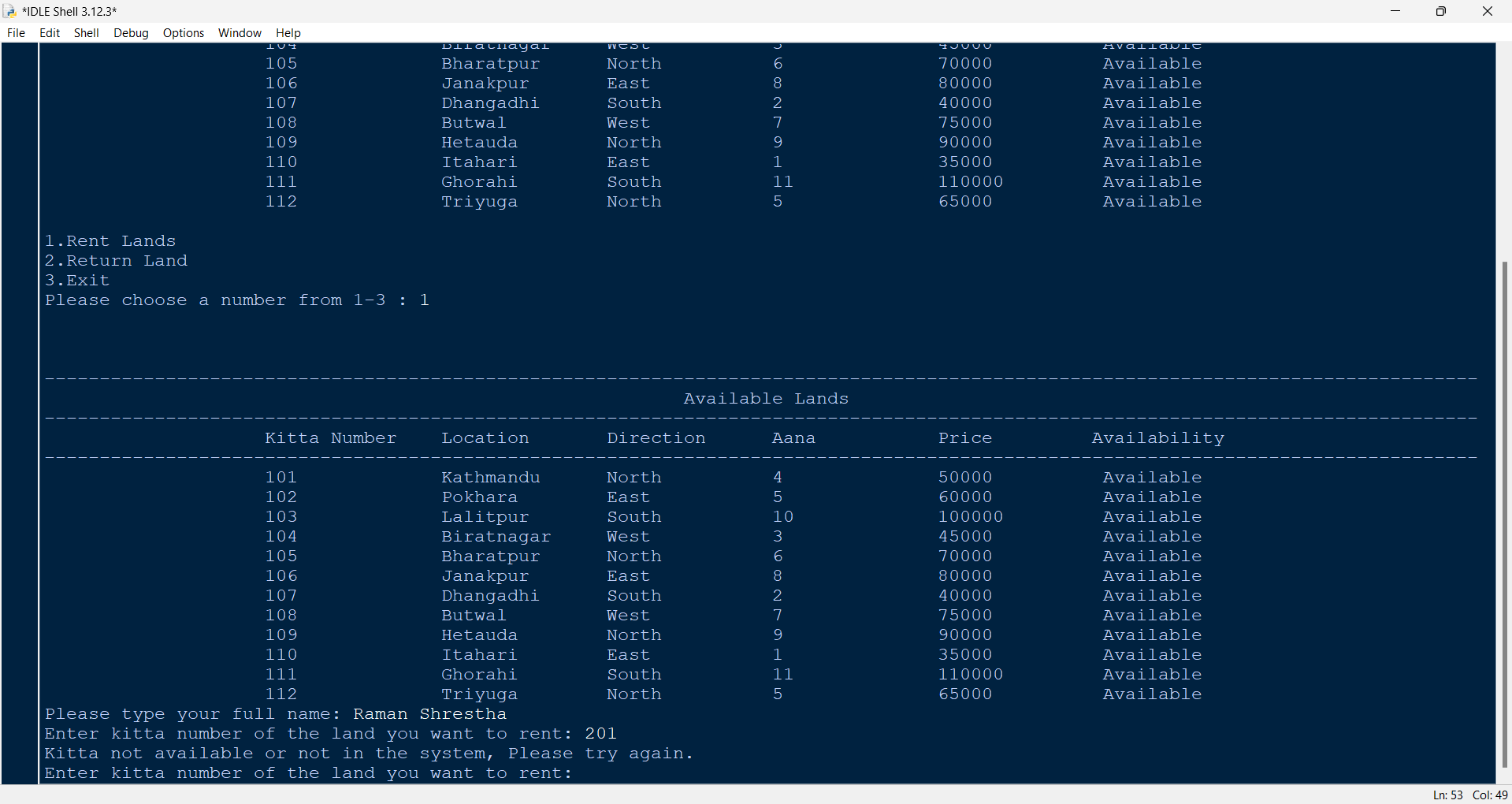


Figure 29: Test -2: To select rent and return of lands - 4

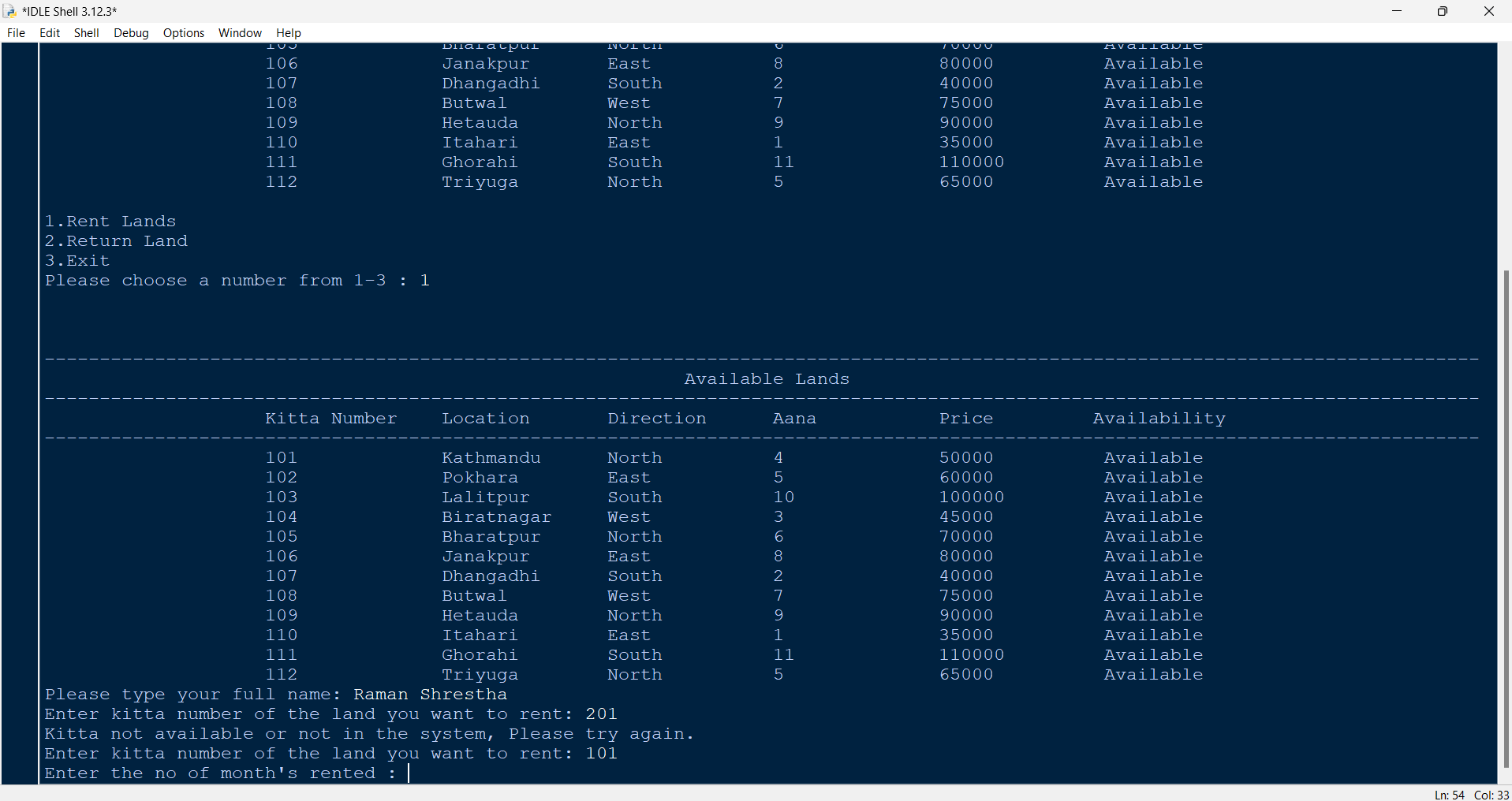


Figure 30: Test -2: To select rent and return of lands - 5

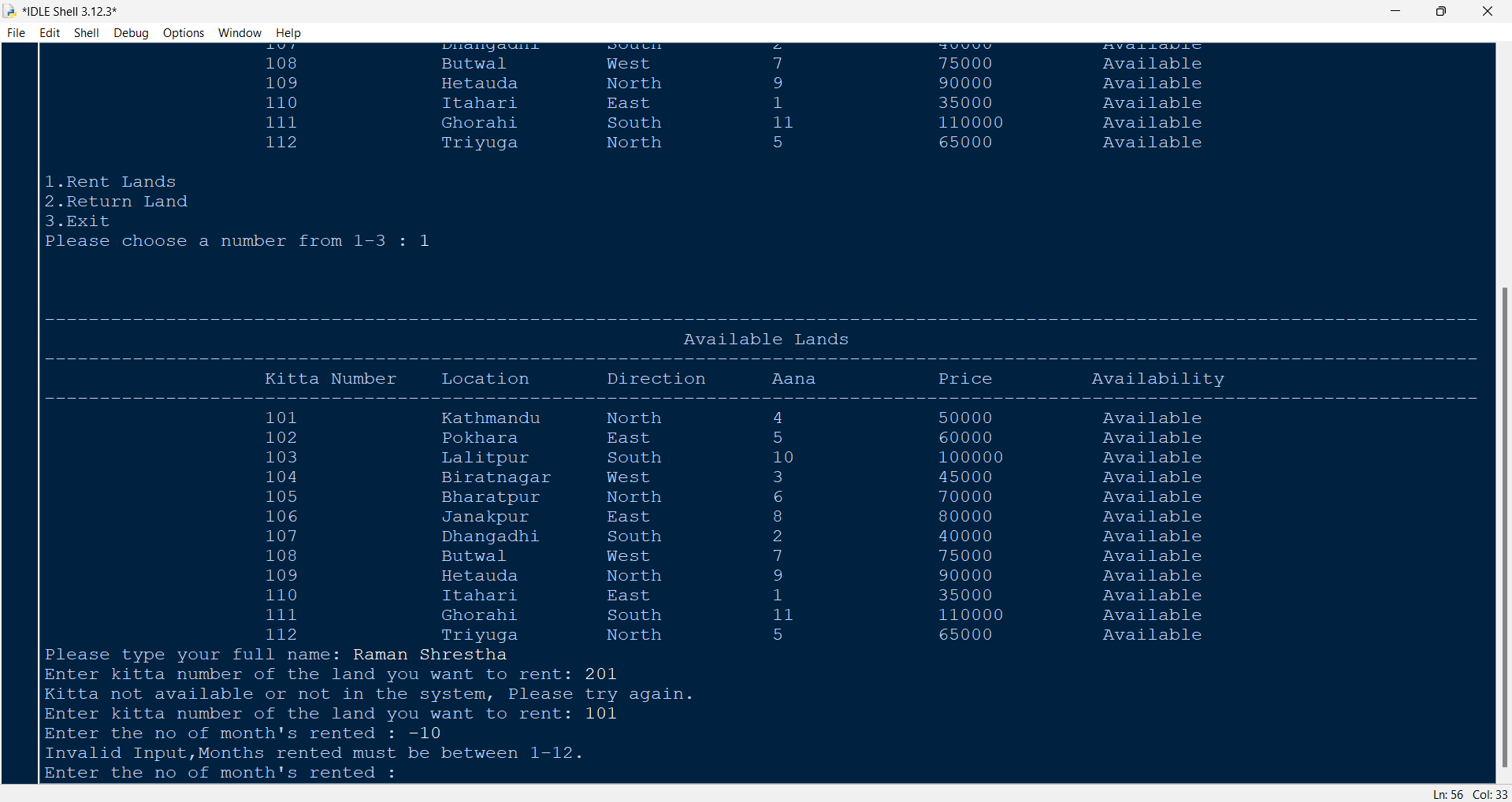


Figure 31: Test -2: To select rent and return of lands - 6

## 4.3 Test – 3: To generate file of renting of land(s)

|  |  |
| --- | --- |
| Test no: | 3 |
| Objective | To generate file of renting of land(s) |
| Action | Run the program  In the home screen, 1 is given as input,  Then, in the name, Hari Shrestha is given as input,  After that, in the kitta number 102 is given as input  In the rented months, 8 is given as input  In the rent again, yes is given as input  Again, in kitta number 101 is given as input,  And in rent 12 is given as input  and in rent again, no is given as input |
| Expected Result | It will update the lands, generate invoice and display it in the screen. |
| Actual Result | It updated the lands, generated the invoice and displayed it in the screen. |
| Conclusion | The test is successful |

Table 3: Test – 3: To generate file of renting of lands

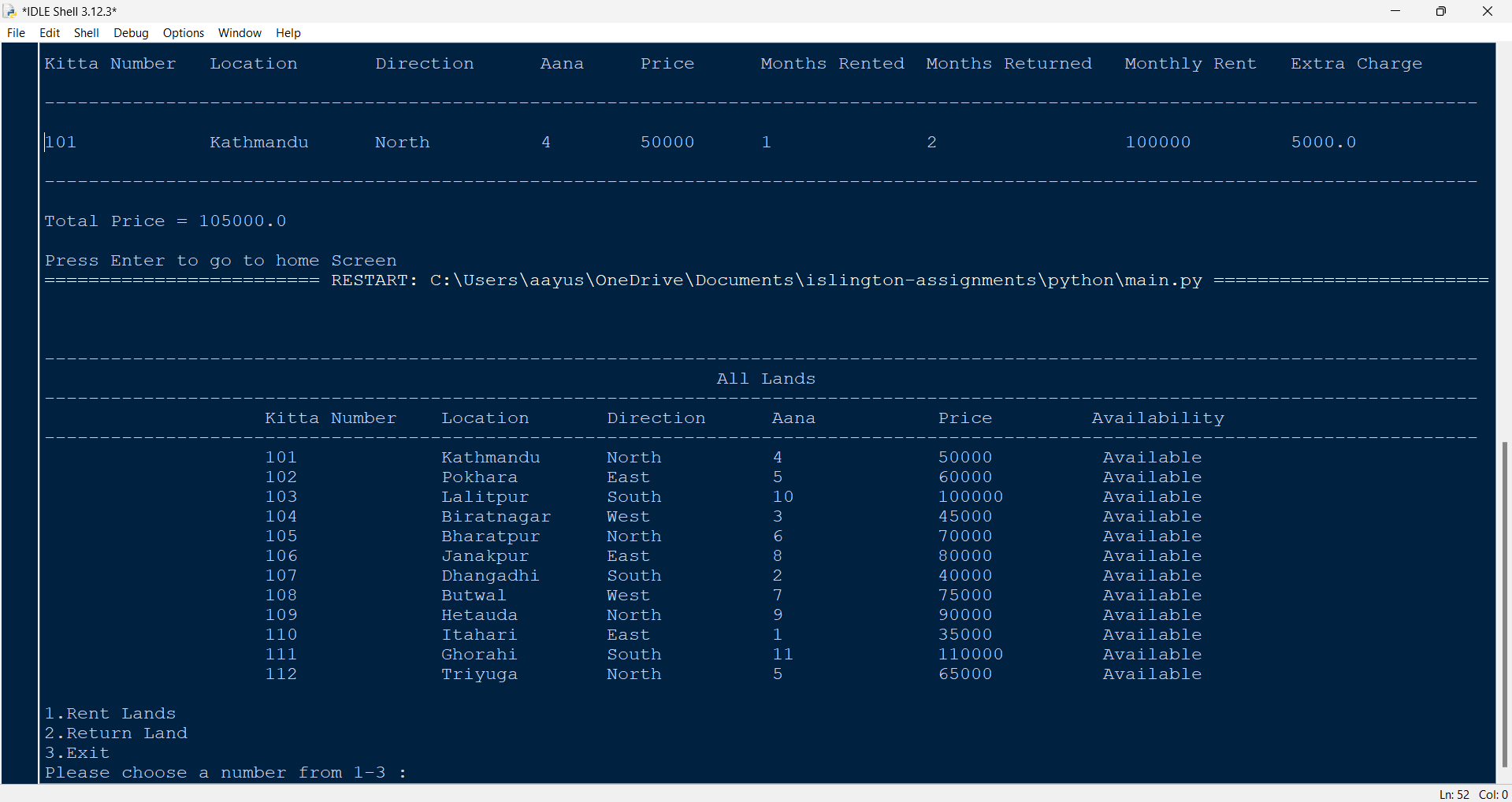


Figure 32: Test – 3 – Homescreen and giving 1 as input

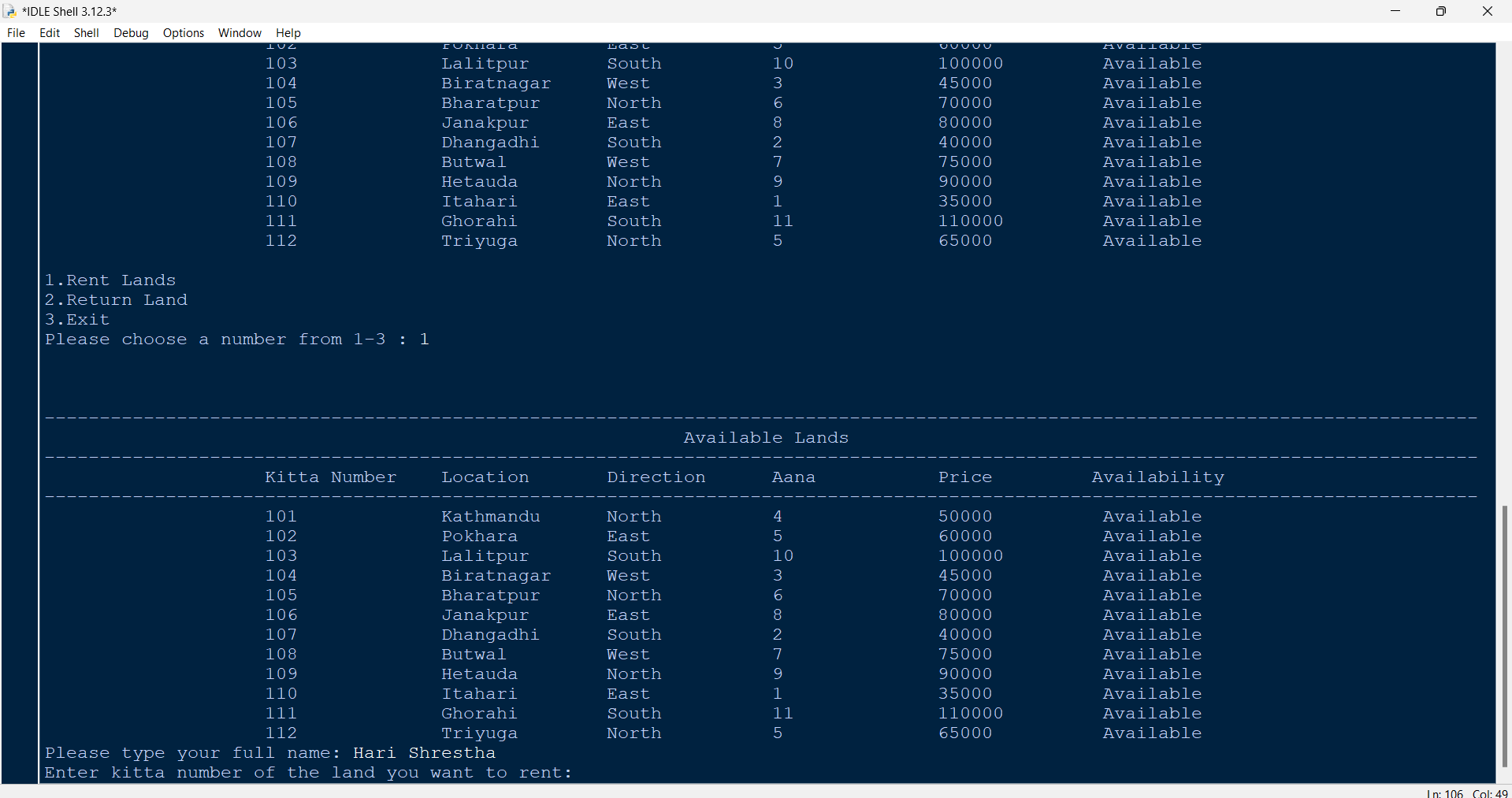


Figure 33: Test – 3 - Giving Hari Shrestha as input in name

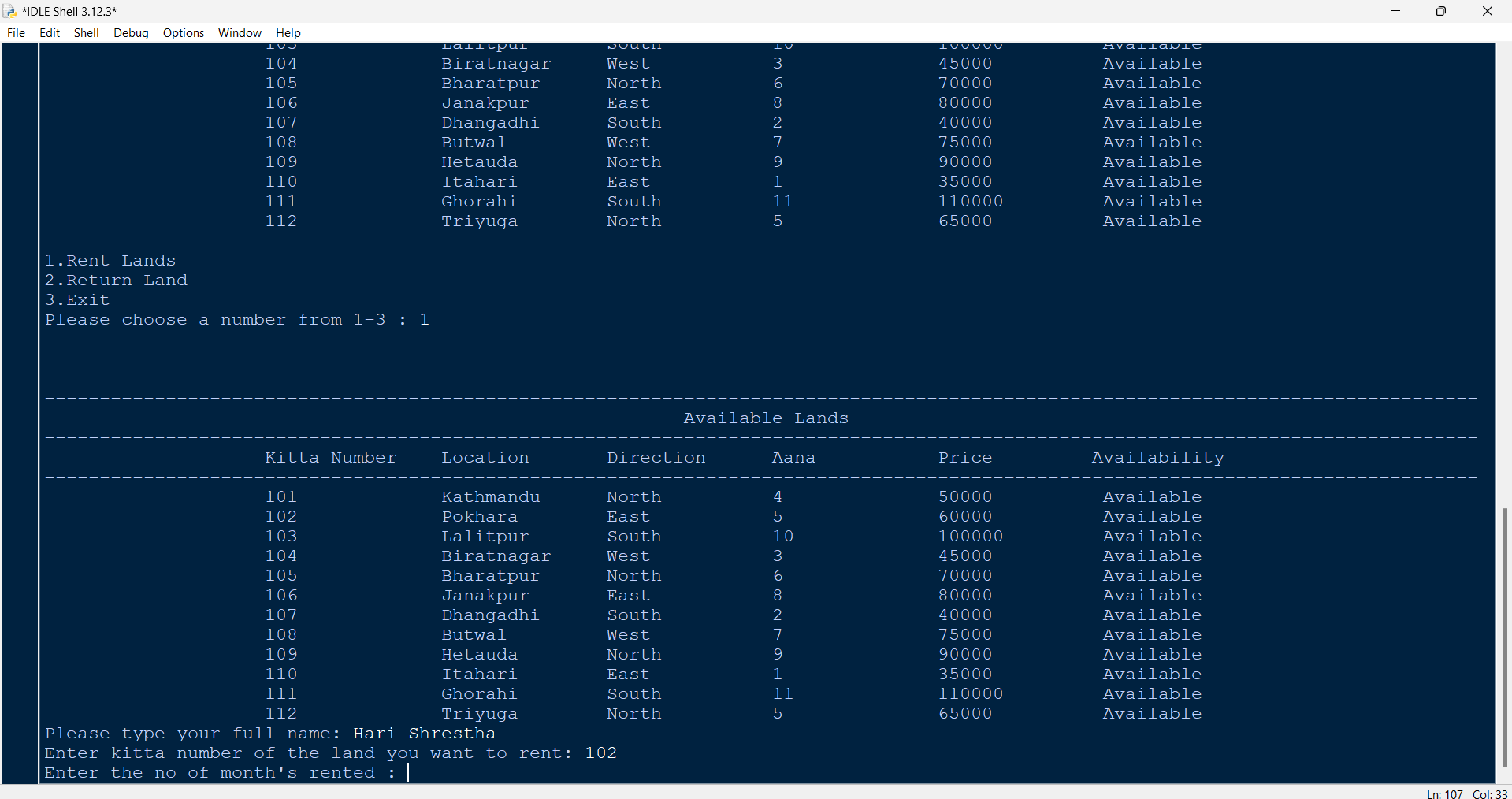


Figure 34: Test – 3 - Giving the kitta number

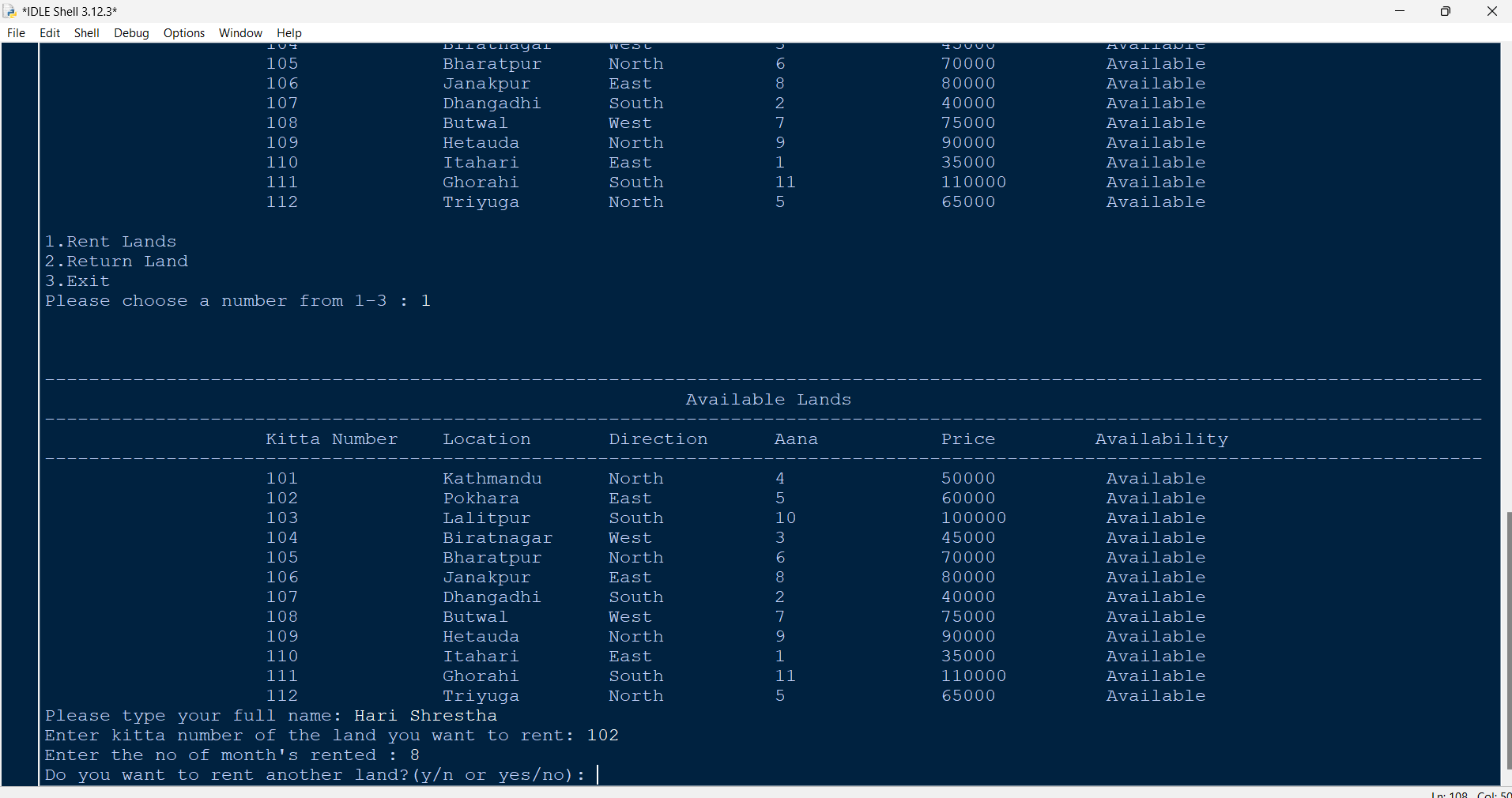


Figure 35: Test – 3: - Giving the months rented

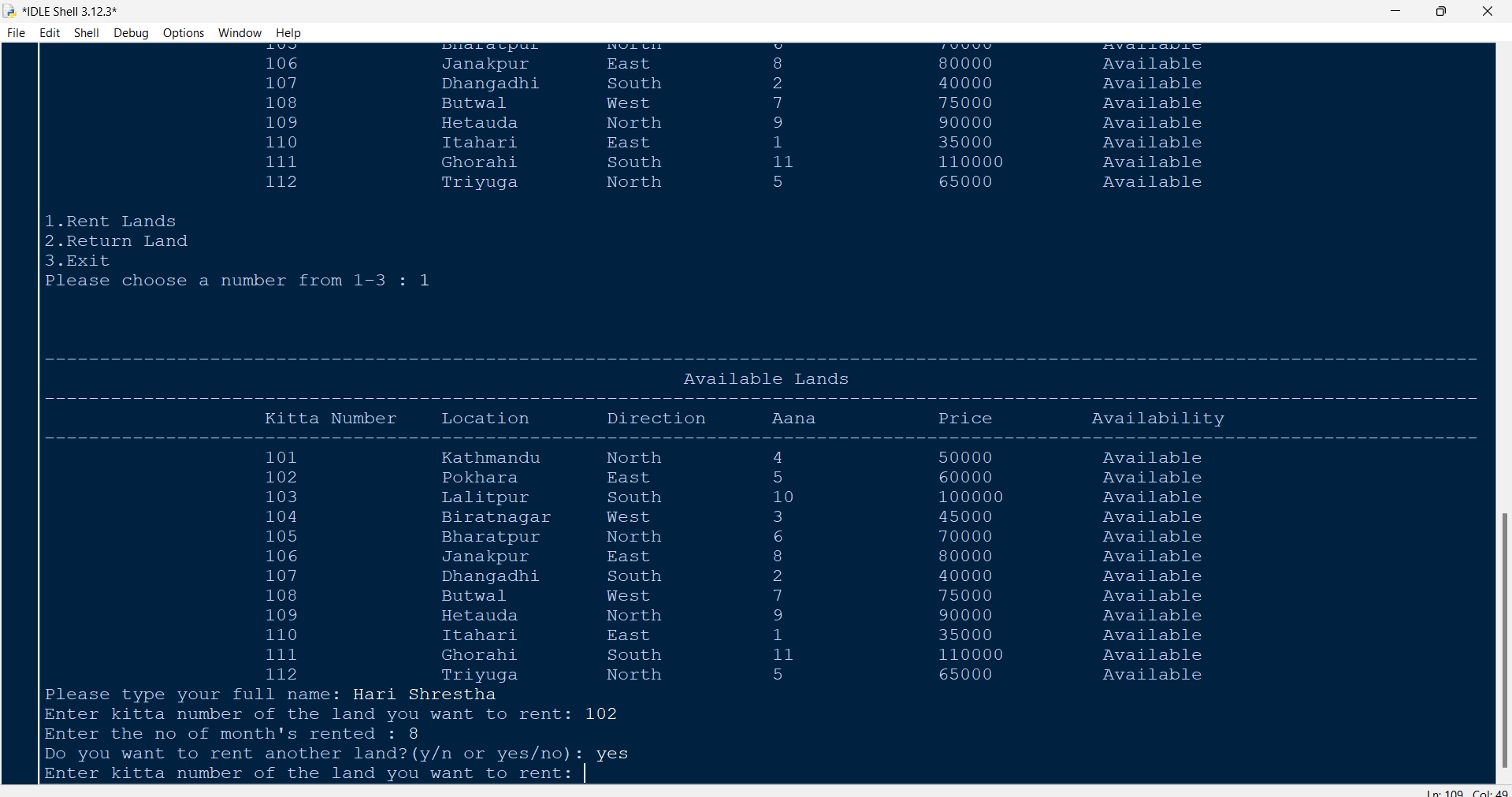


Figure 36: Test – 3 - Giving the rent another land yes

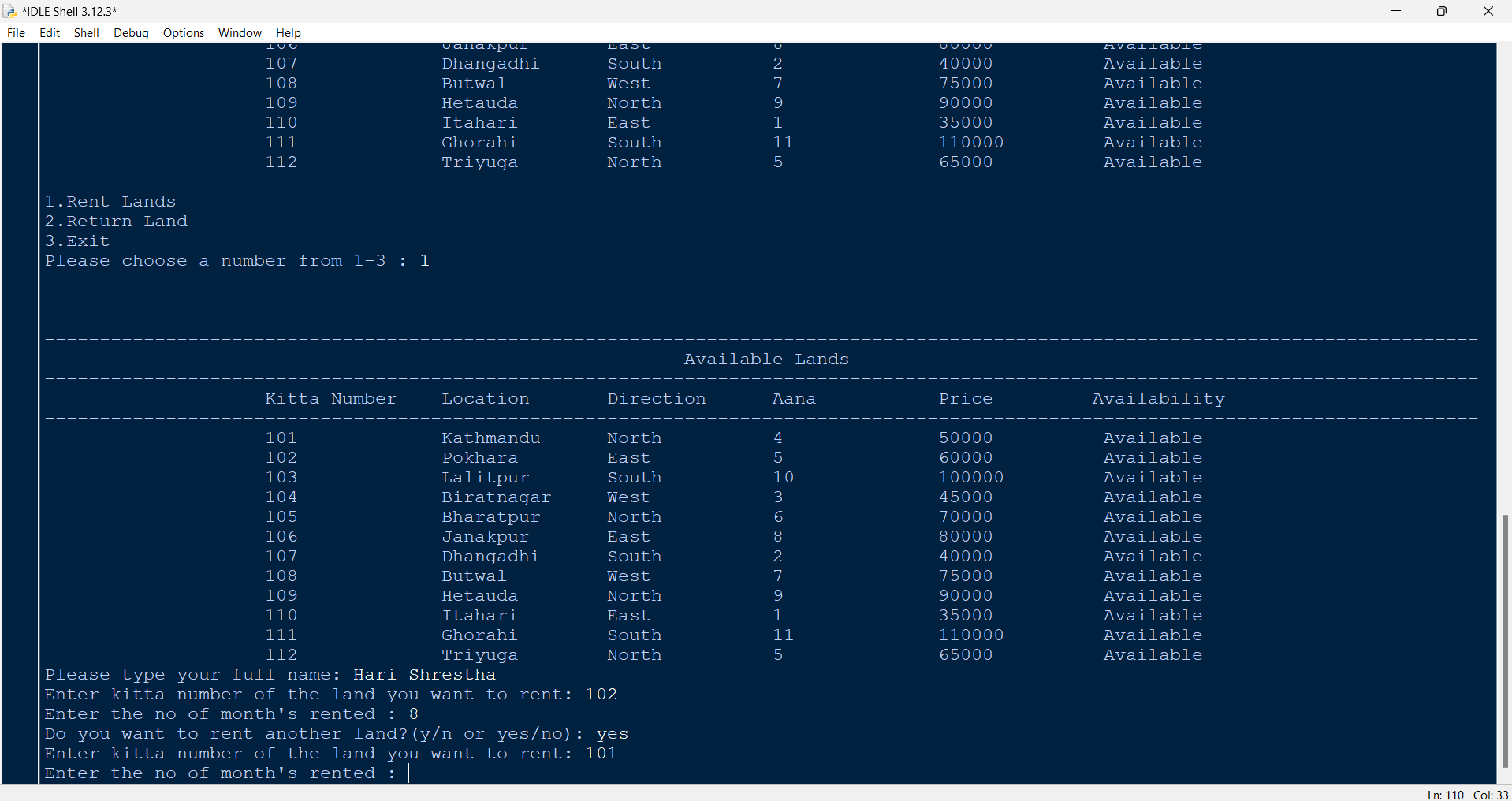


Figure 37: Test – 3 - Giving kitta number 101 as input in kitta

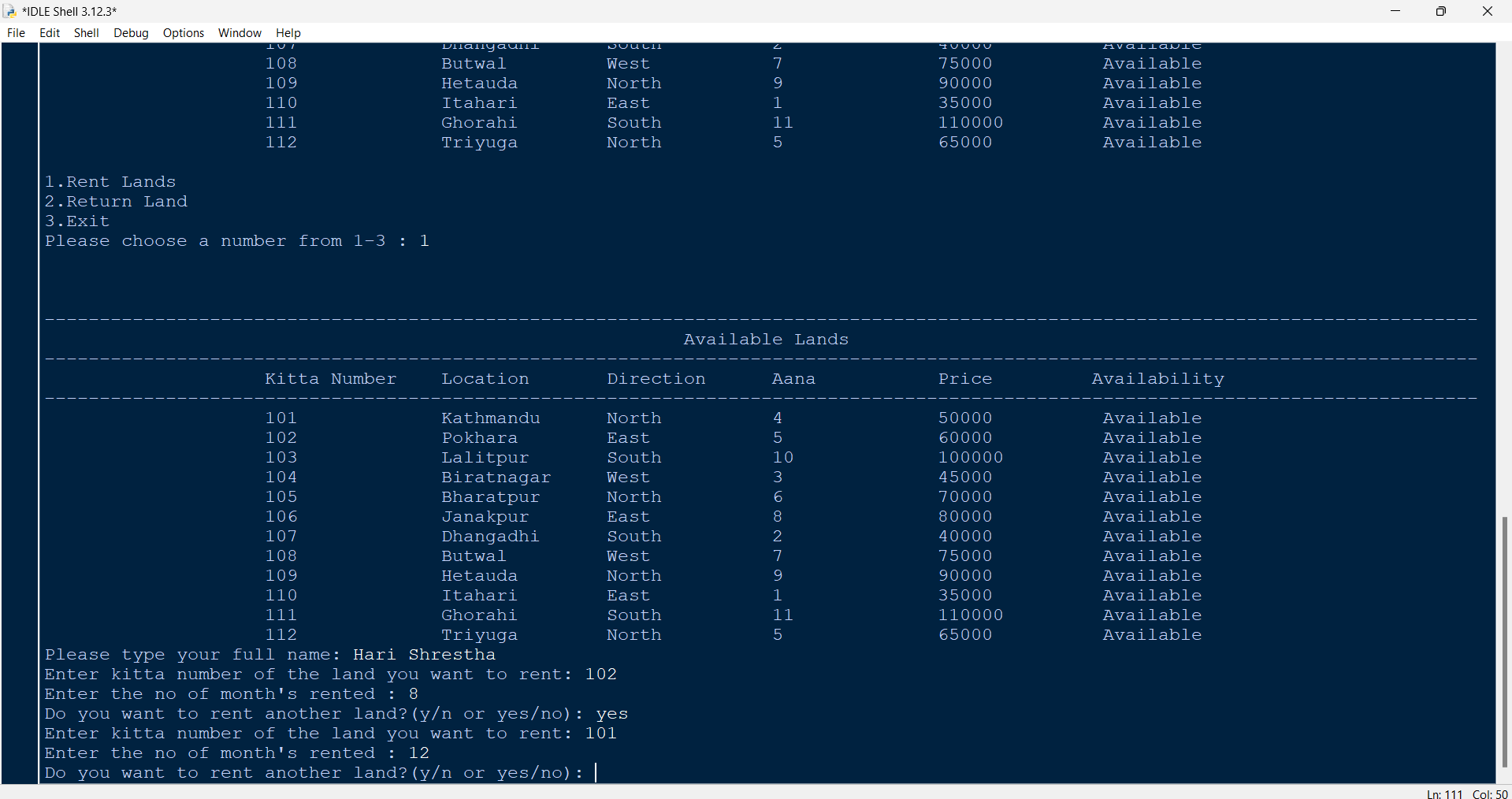


Figure 38: Test – 3 - Giving month as input

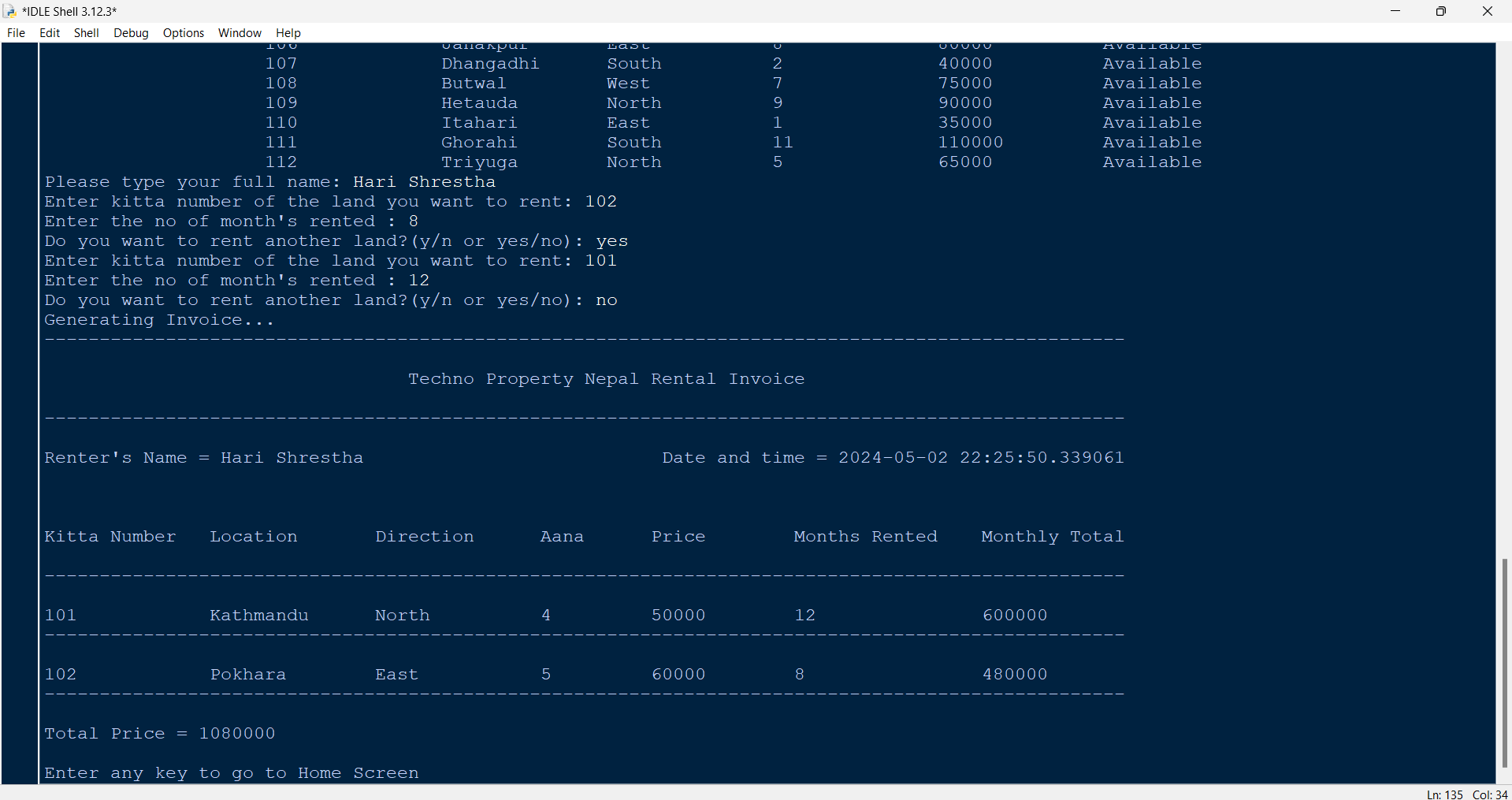


Figure 39: Test – 3 - Giving no as input in rent more and invoice generation

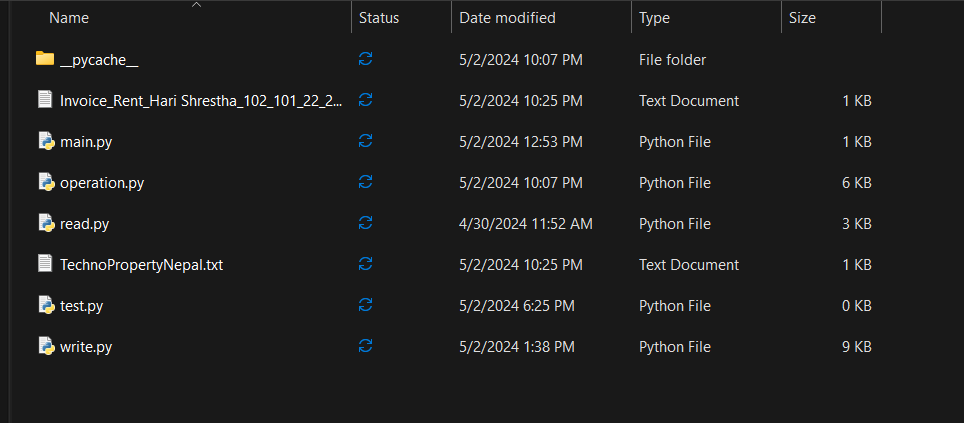


Figure 40: Test – 3 - Invoice generated location

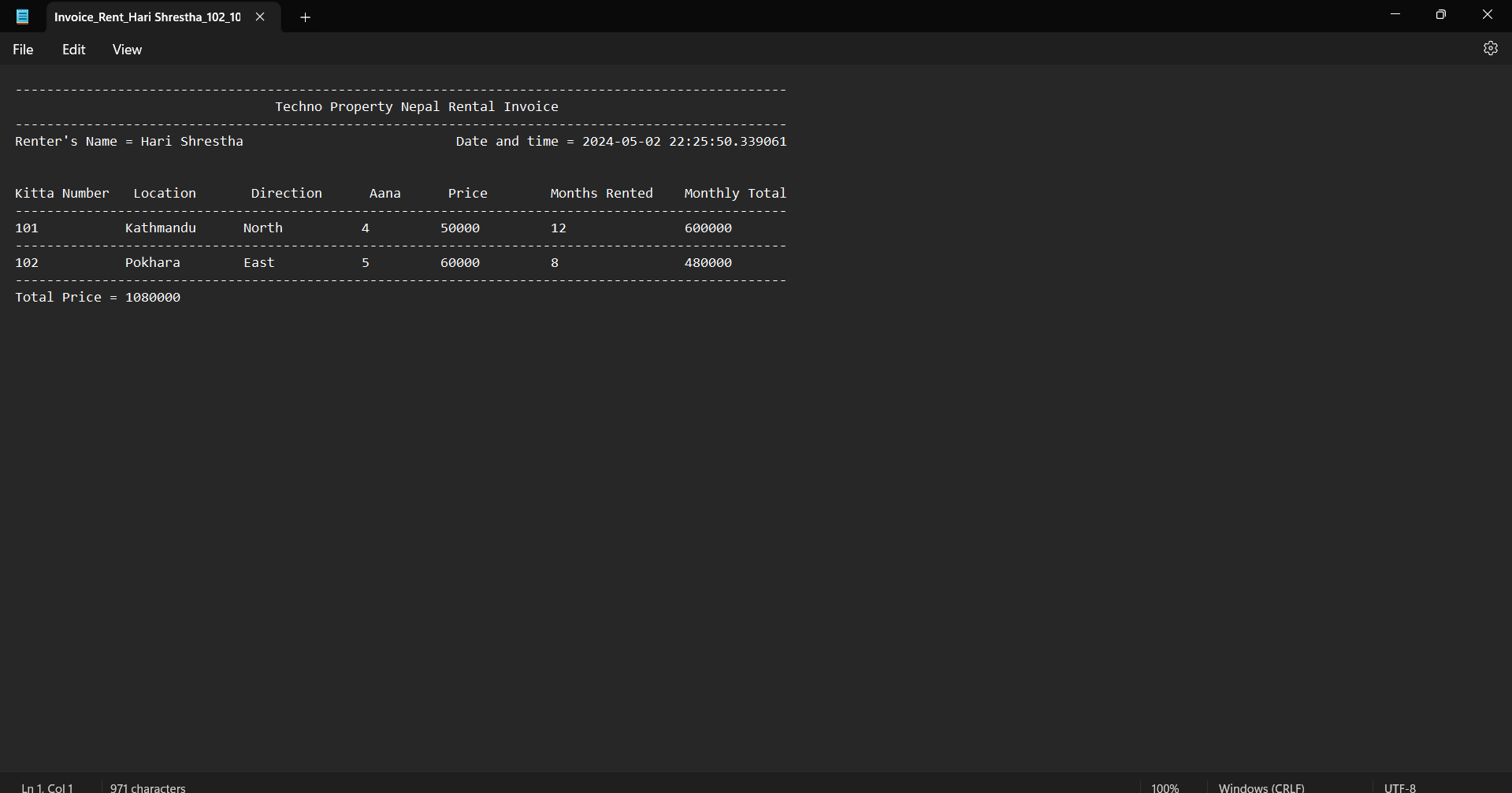


Figure 41: Test – 3 - Renting invoice

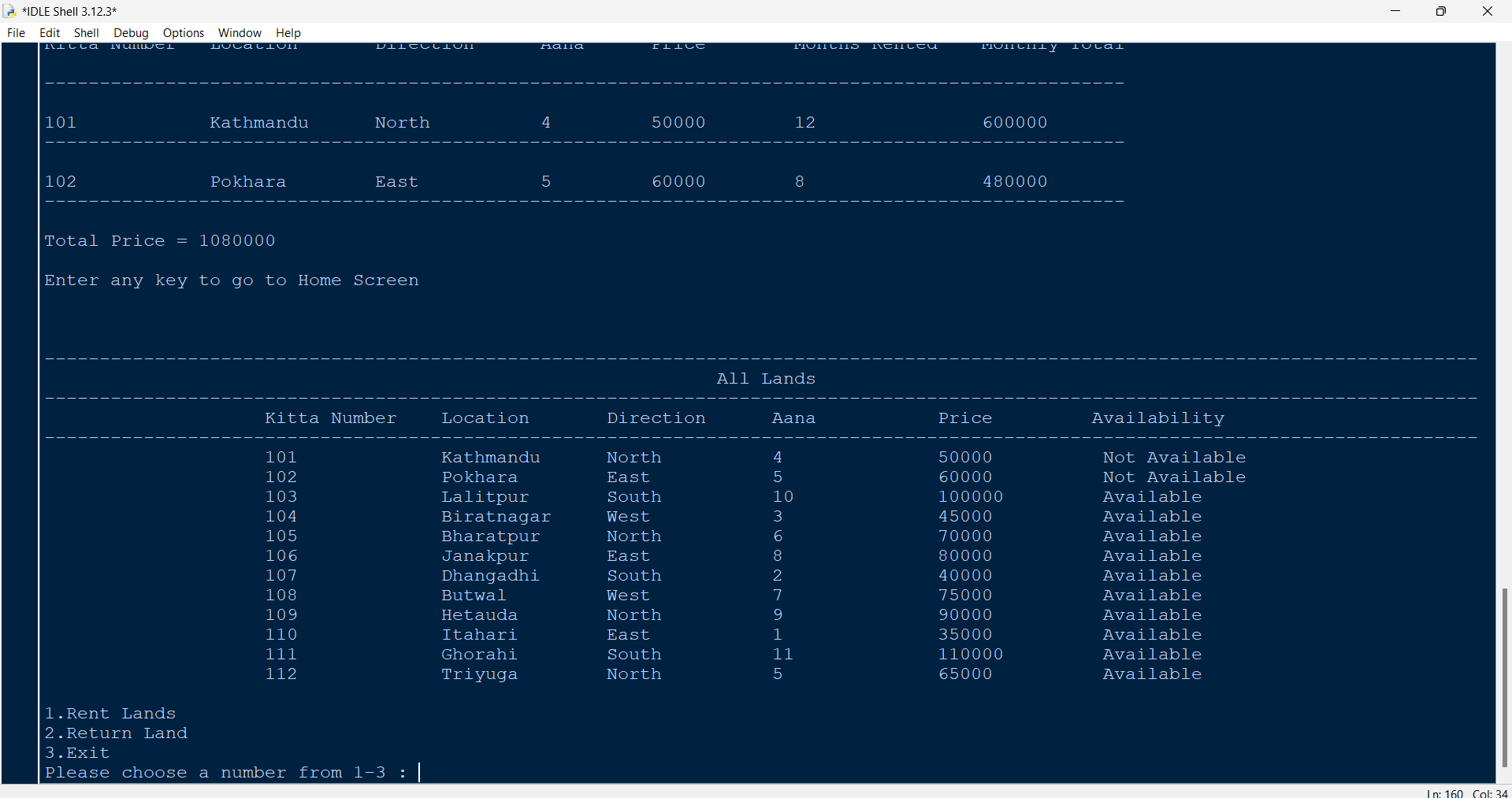


Figure 42: Test – 3 - Updated lands in home screen



Figure 43: Test – 3 - Updated lands text file

## 4.4 Test – 4: To generate file of returning the land(s)

|  |  |
| --- | --- |
| Test no: | 4 |
| Objective | To generate file of returning the land(s) |
| Action | Run the program  In the home screen, 2 is given as input,  Then, in the name, Hari Shrestha is given as input,  After that, in the kitta number 101 is given as input  In the rented months, 8 is given as input  In the returned months, 5 is given as input  In the return again, yes is given as input  Again, in kitta number 102 is given as input,  And in rent 12 is given as input  In the returned month, 20 is given as input  and in return again, no is given as input |
| Expected Result | It will update the lands, generate the return invoice and display it in the screen. |
| Actual Result | It updated the lands, generated the return invoice and displayed it in the screen. |
| Conclusion | The test is successful |

Table 4: Test – 4: To generate file of returning the land(s)

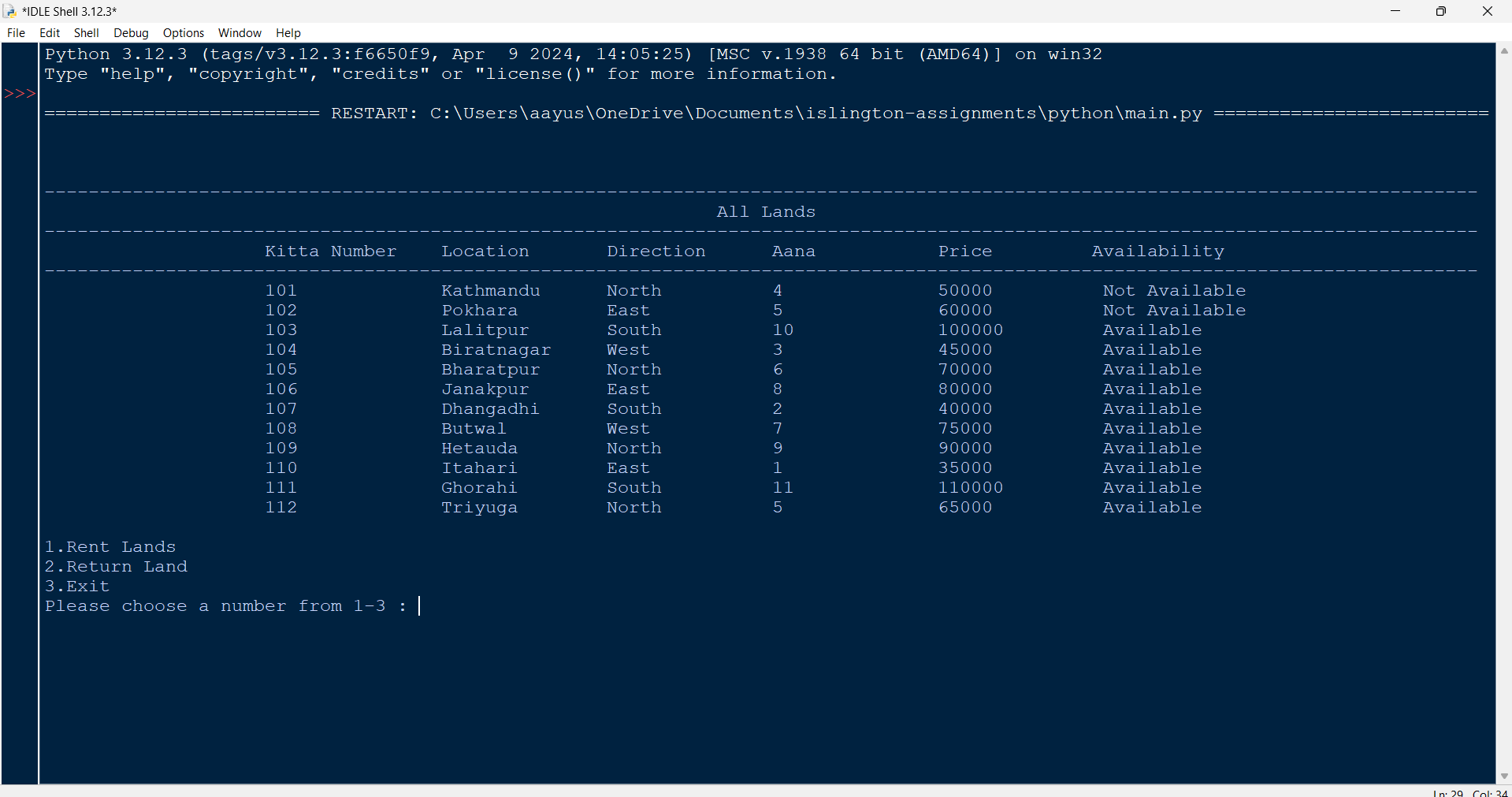


Figure 44: Test – 4 - Home screen and giving 2 as input

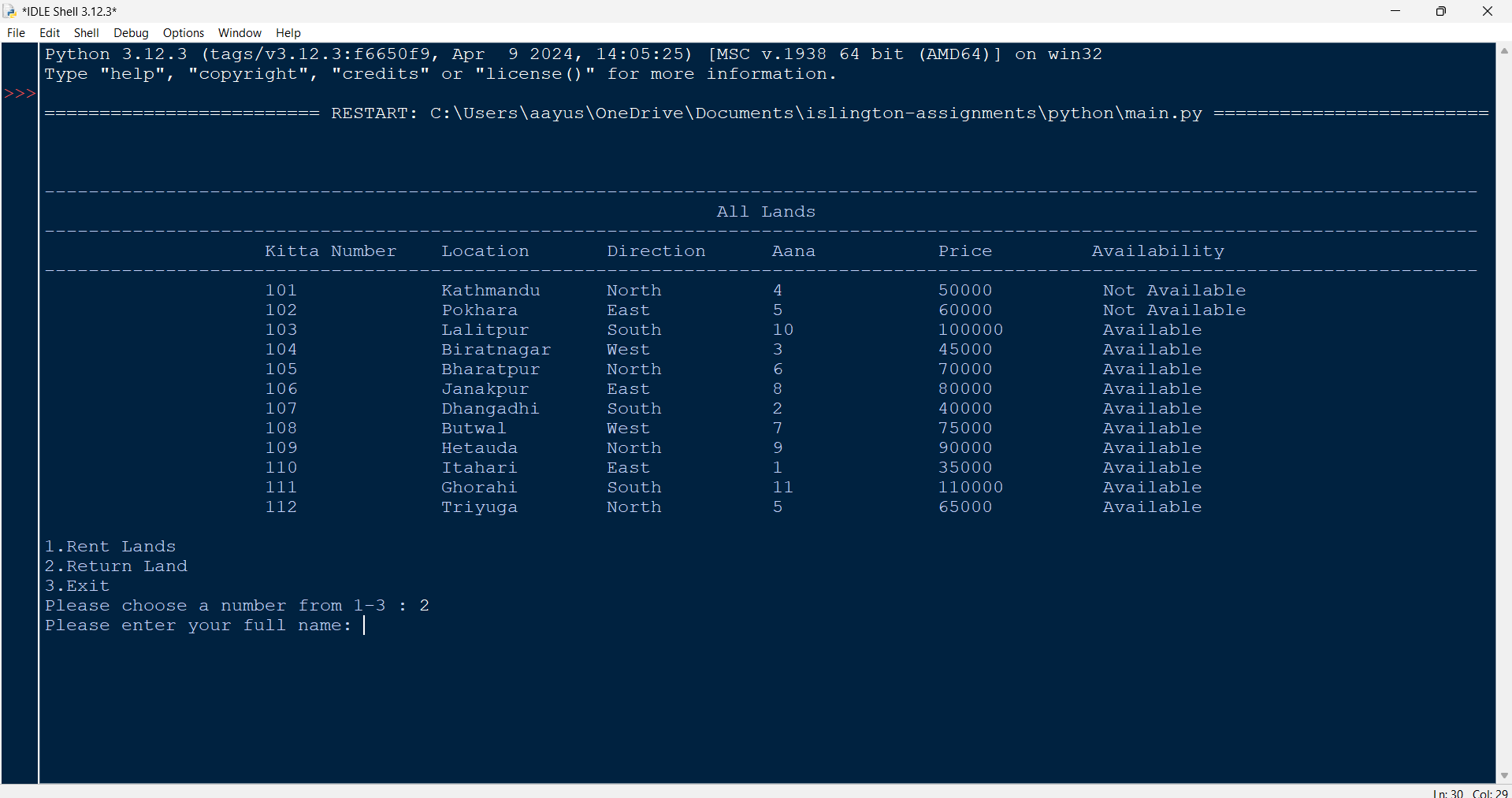


Figure 45: Test – 4 – Giving name as input

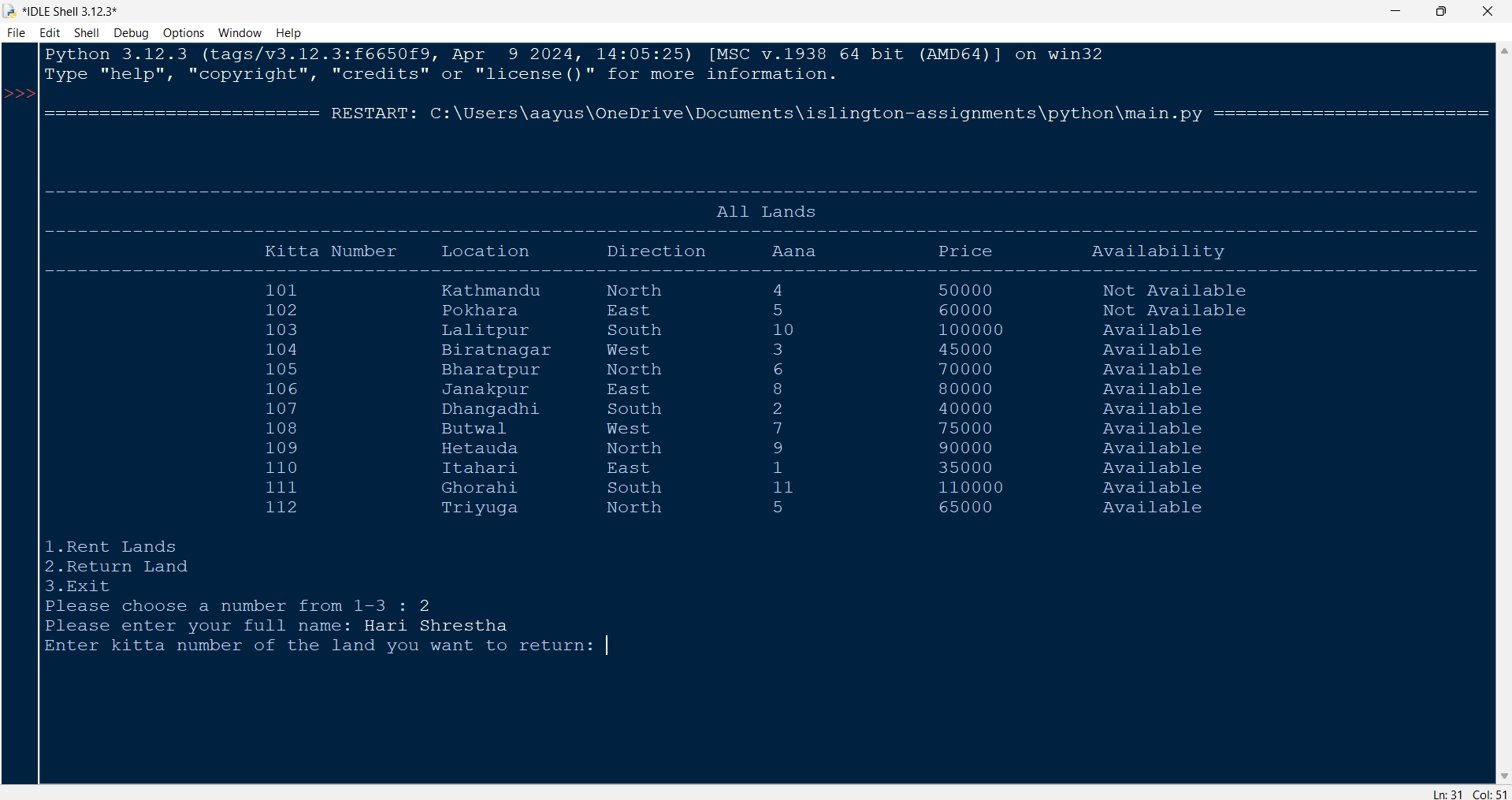


Figure 46: Test – 4 - Giving kitta numbers as input

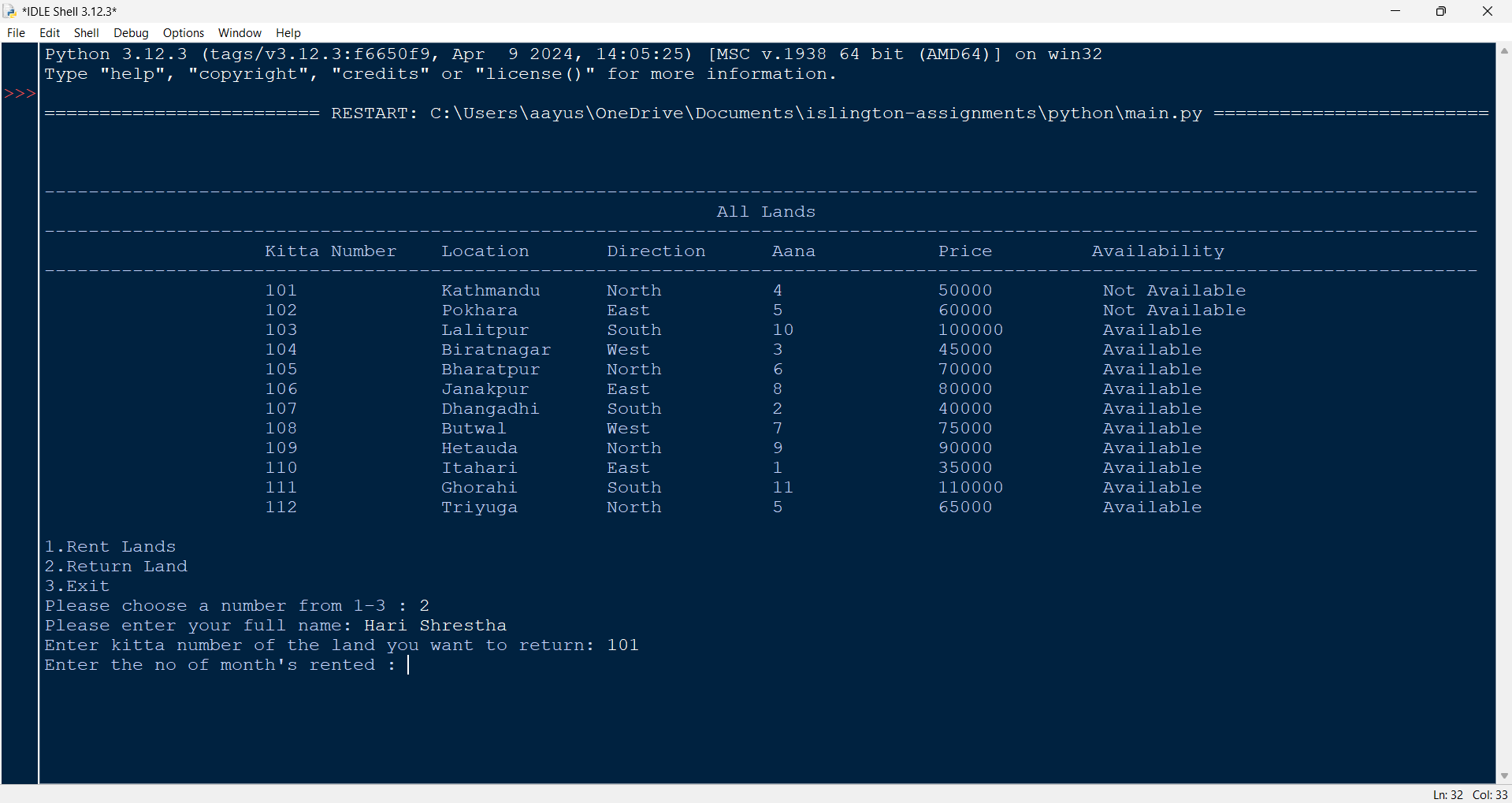


Figure 47: Test – 4 - Giving month rented as input

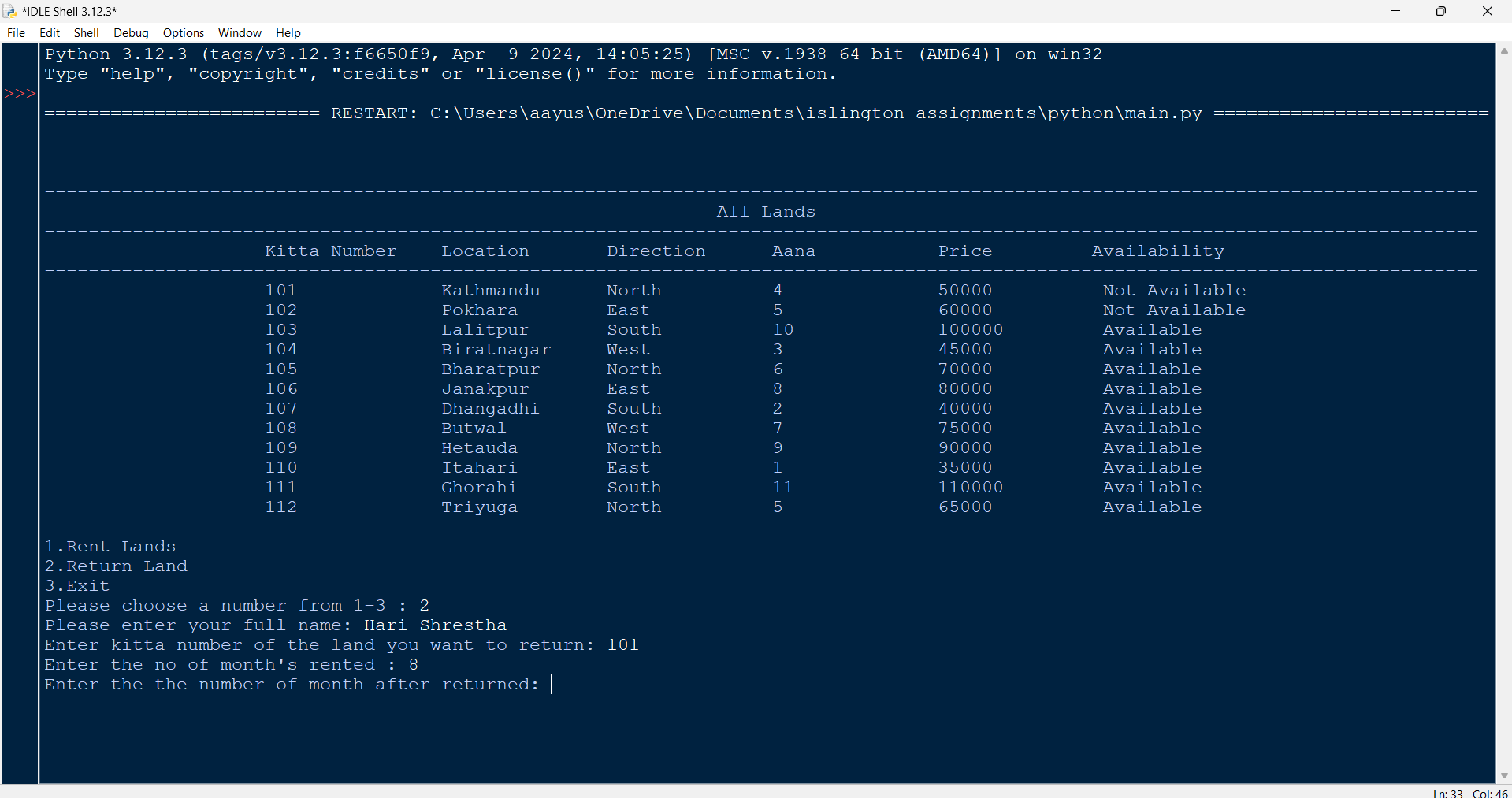


Figure 48: Test – 4 - Giving returned month as input

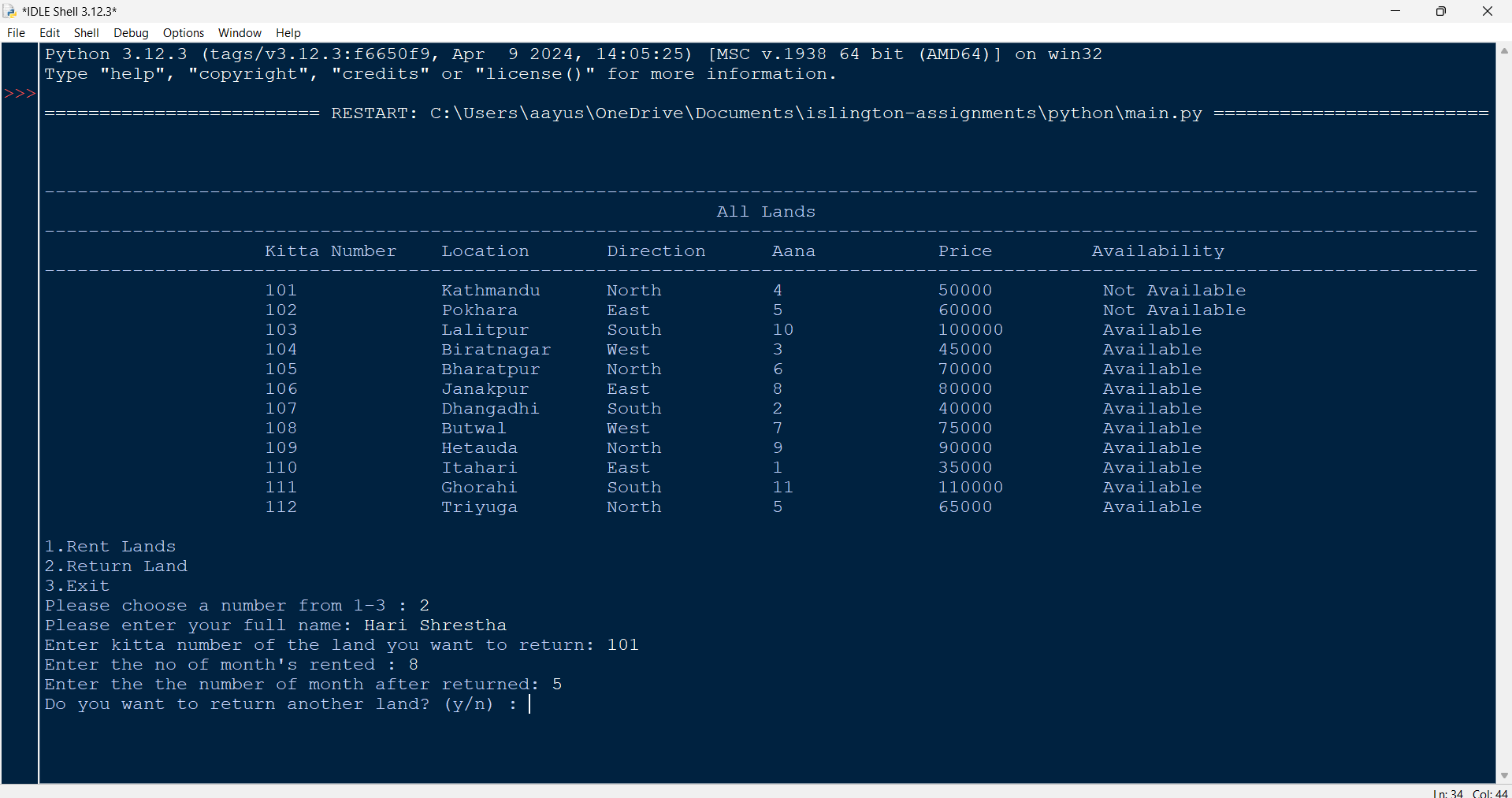


Figure 49: Test – 4 - Giving yes as input to return more land

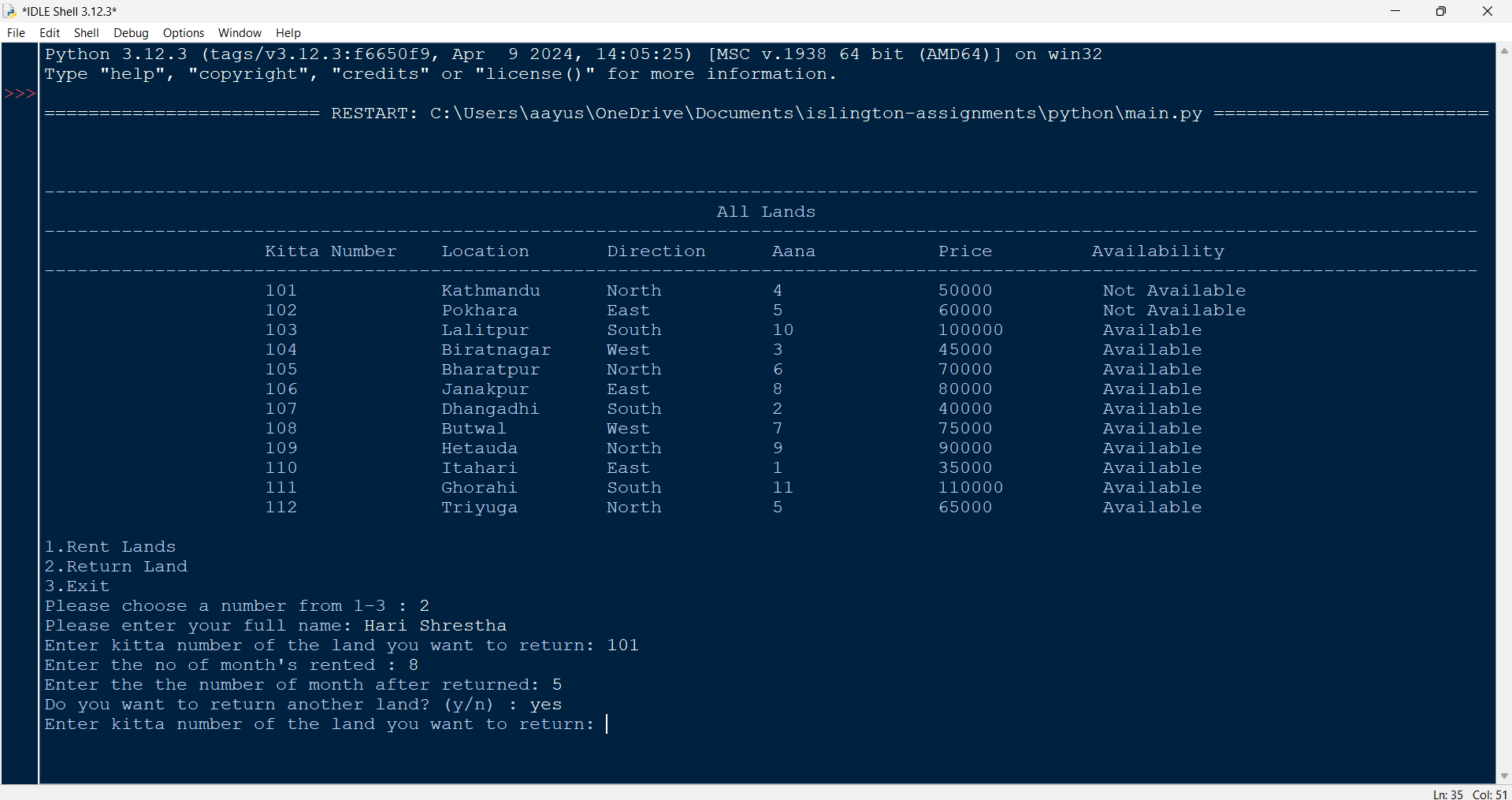


Figure 50: Test – 4 - Giving second kitta number as input

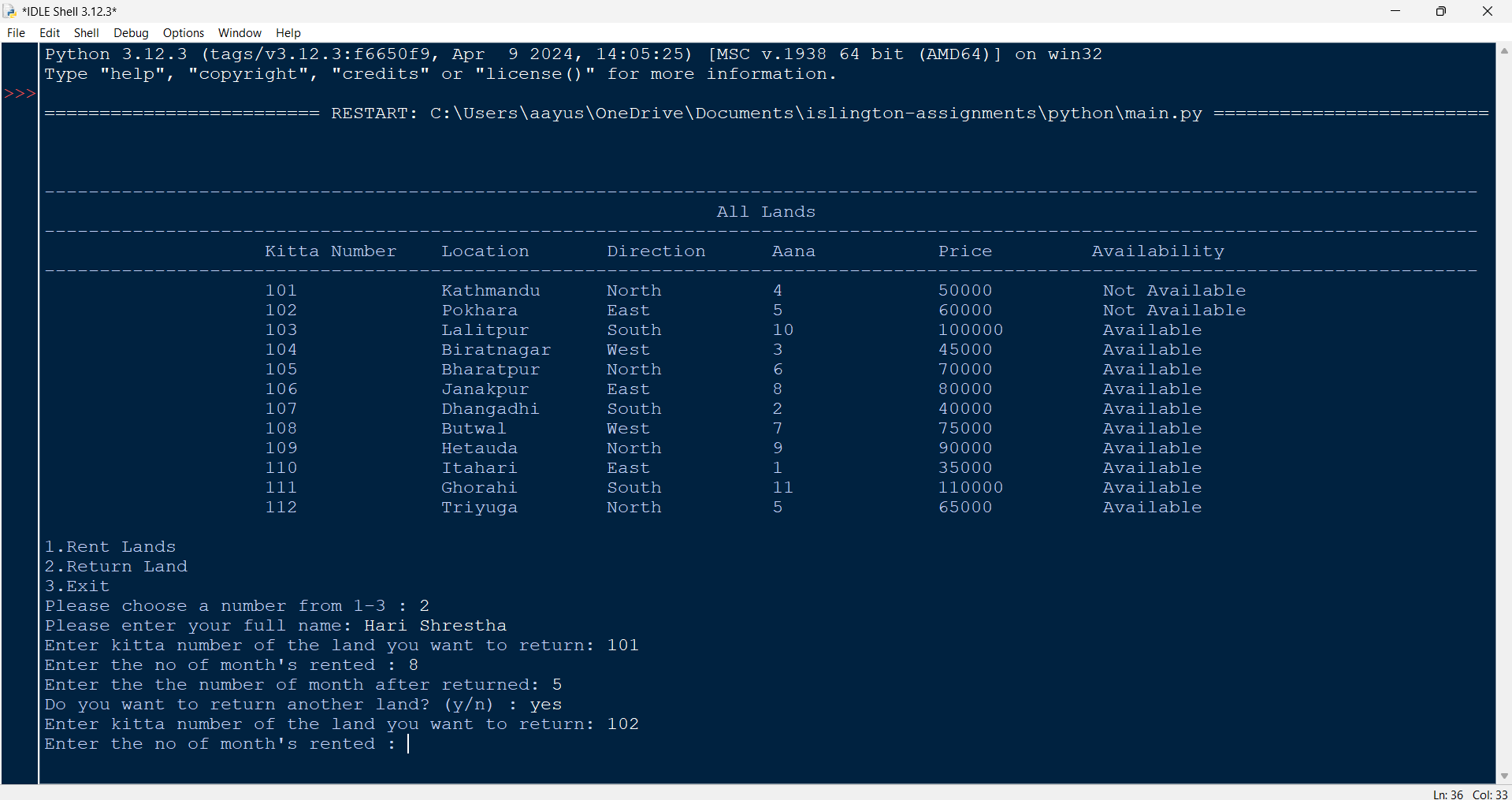


Figure 51: Test – 4 - Giving no of months rented

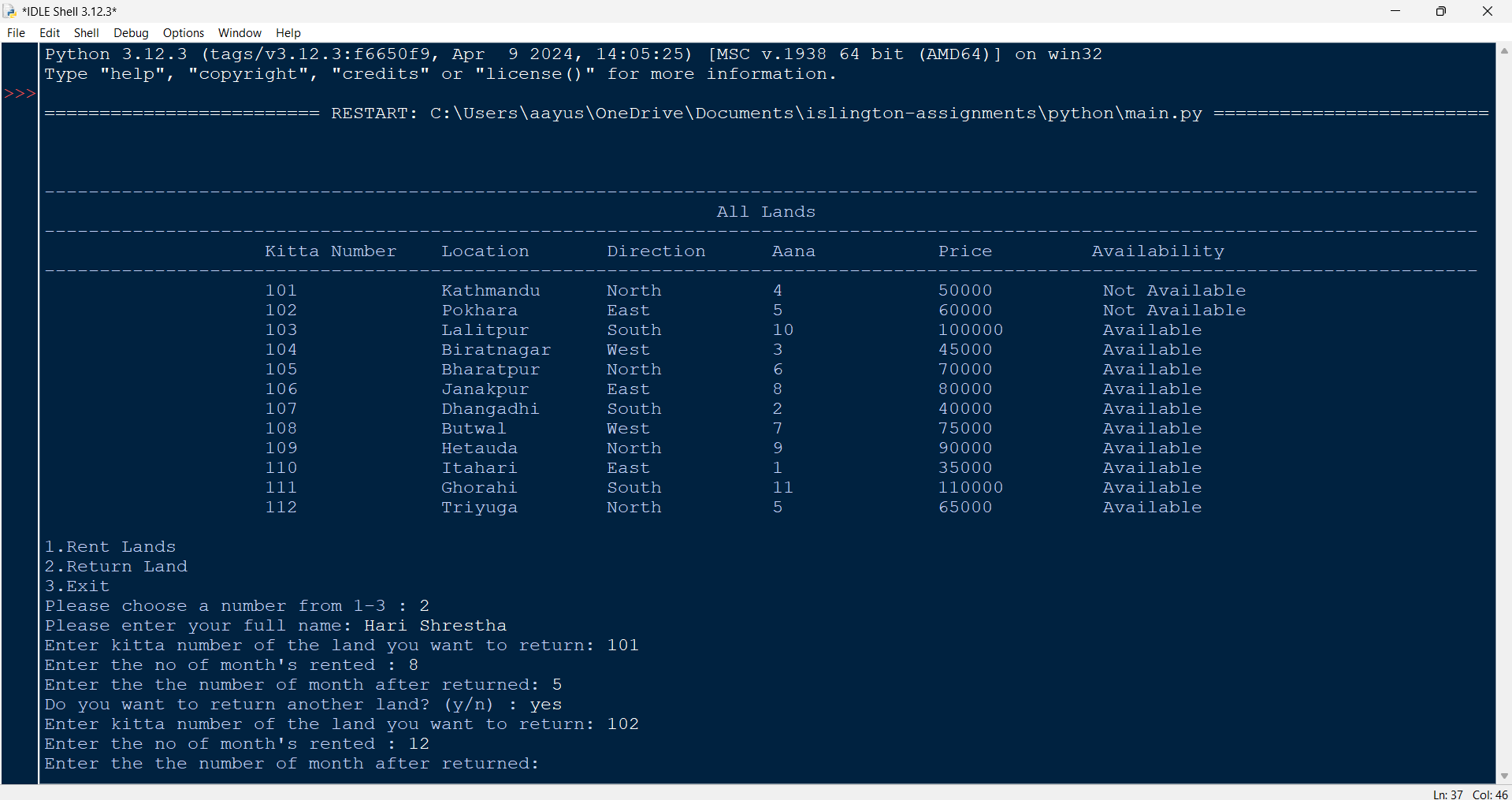


Figure 52: Test – 4 - Giving returned month as input

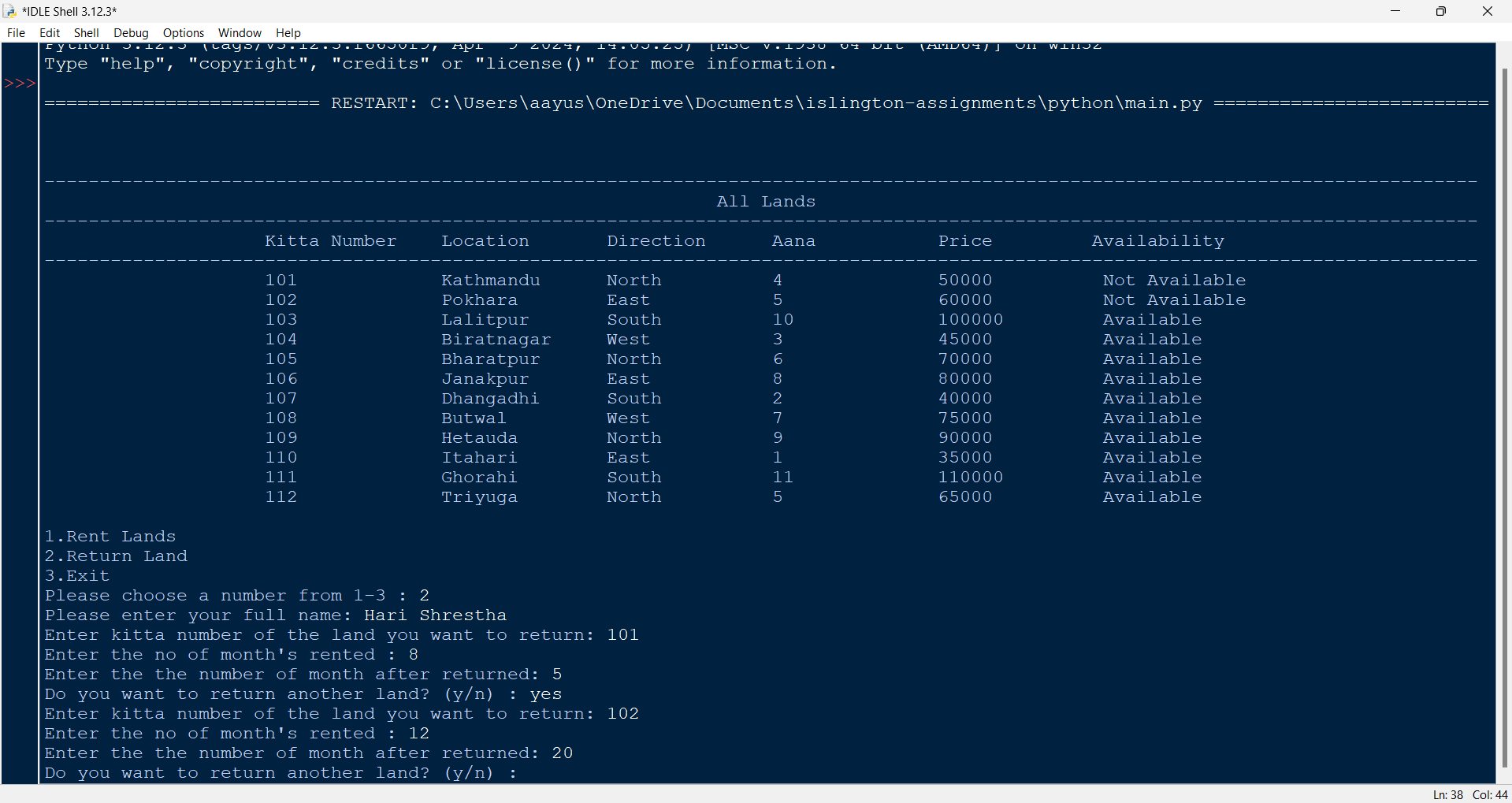


Figure 53: Test – 4 - Giving no as input to return more land

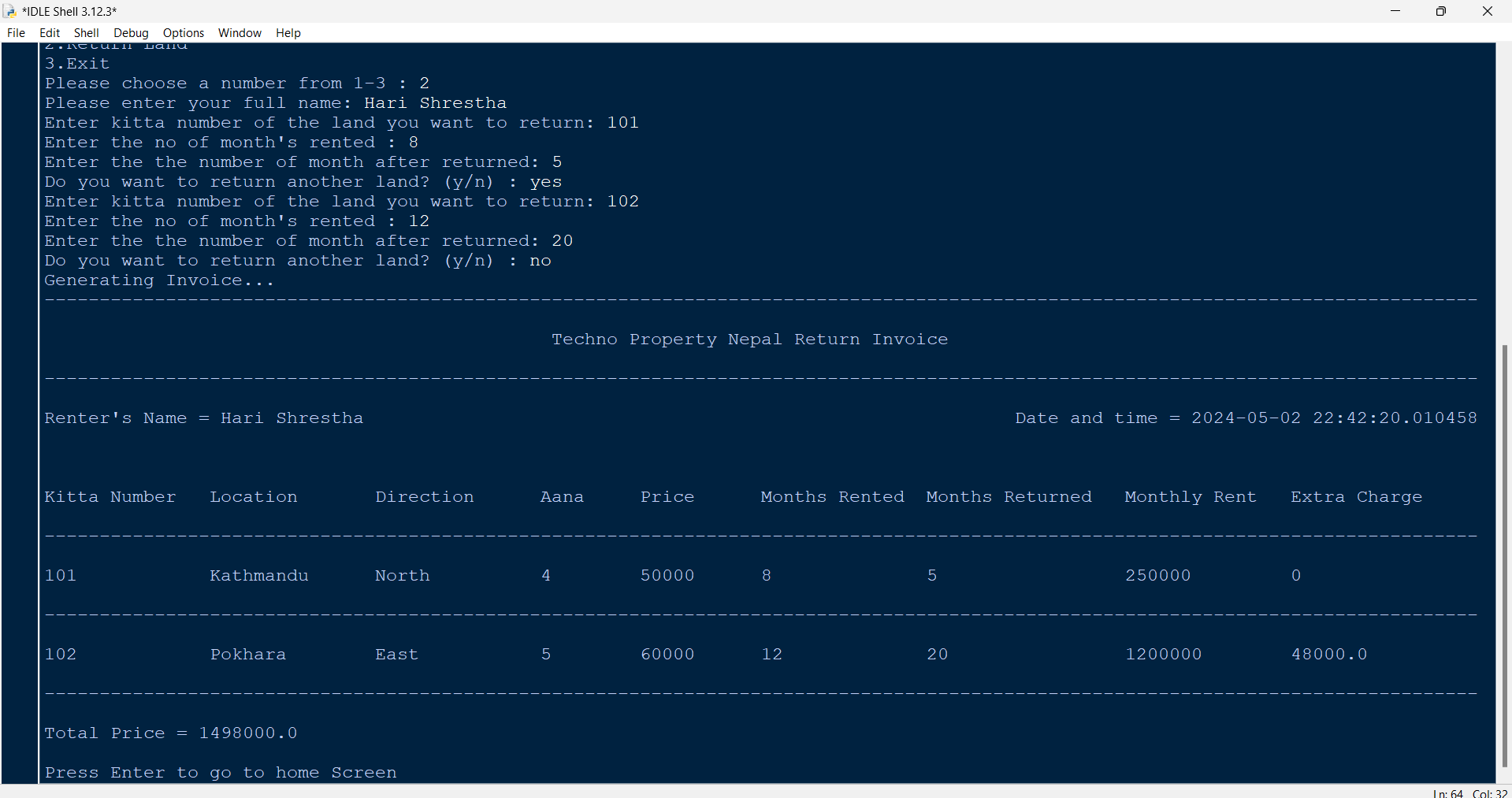


Figure 54: Test – 4 - Generating and displaying invoice

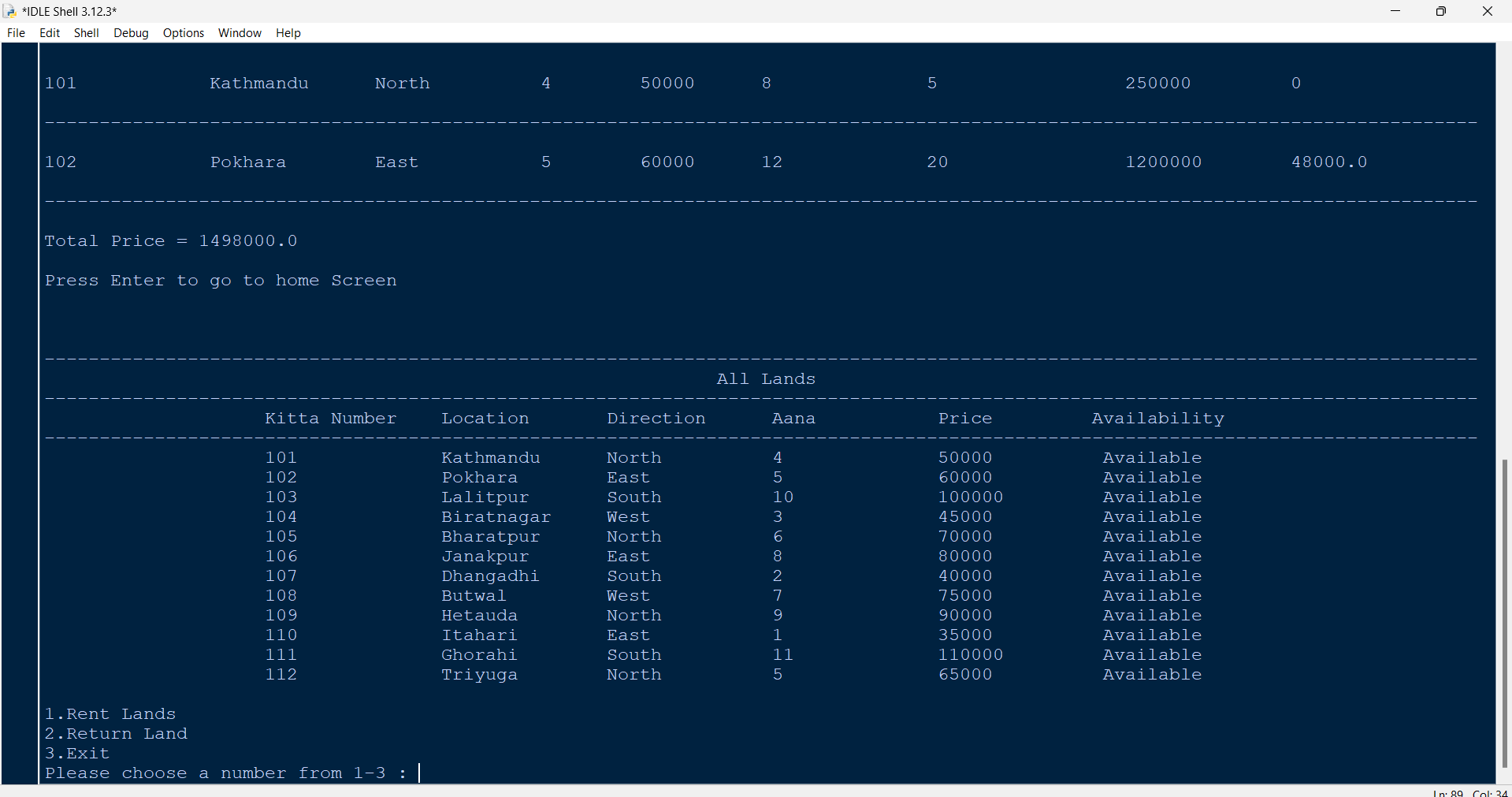


Figure 55: Test – 4 - Updated land availability in the cell

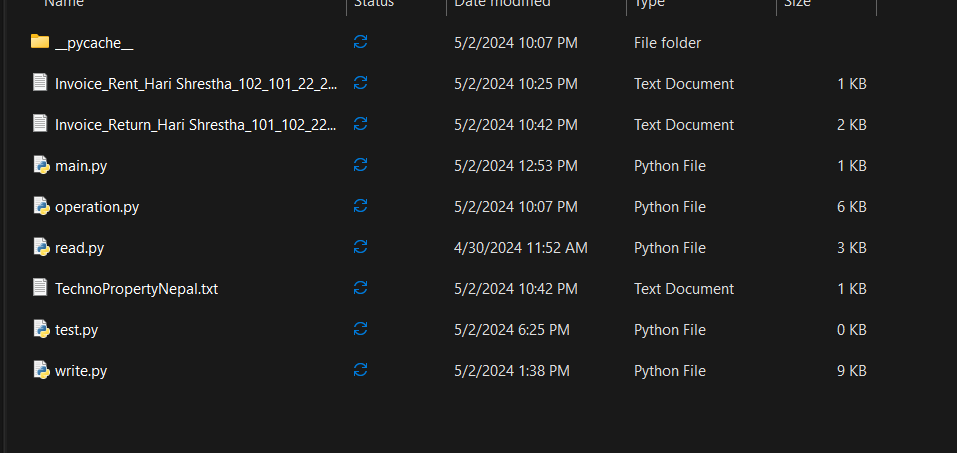


Figure 56: Test – 4 - Generation of return invoice in the folder

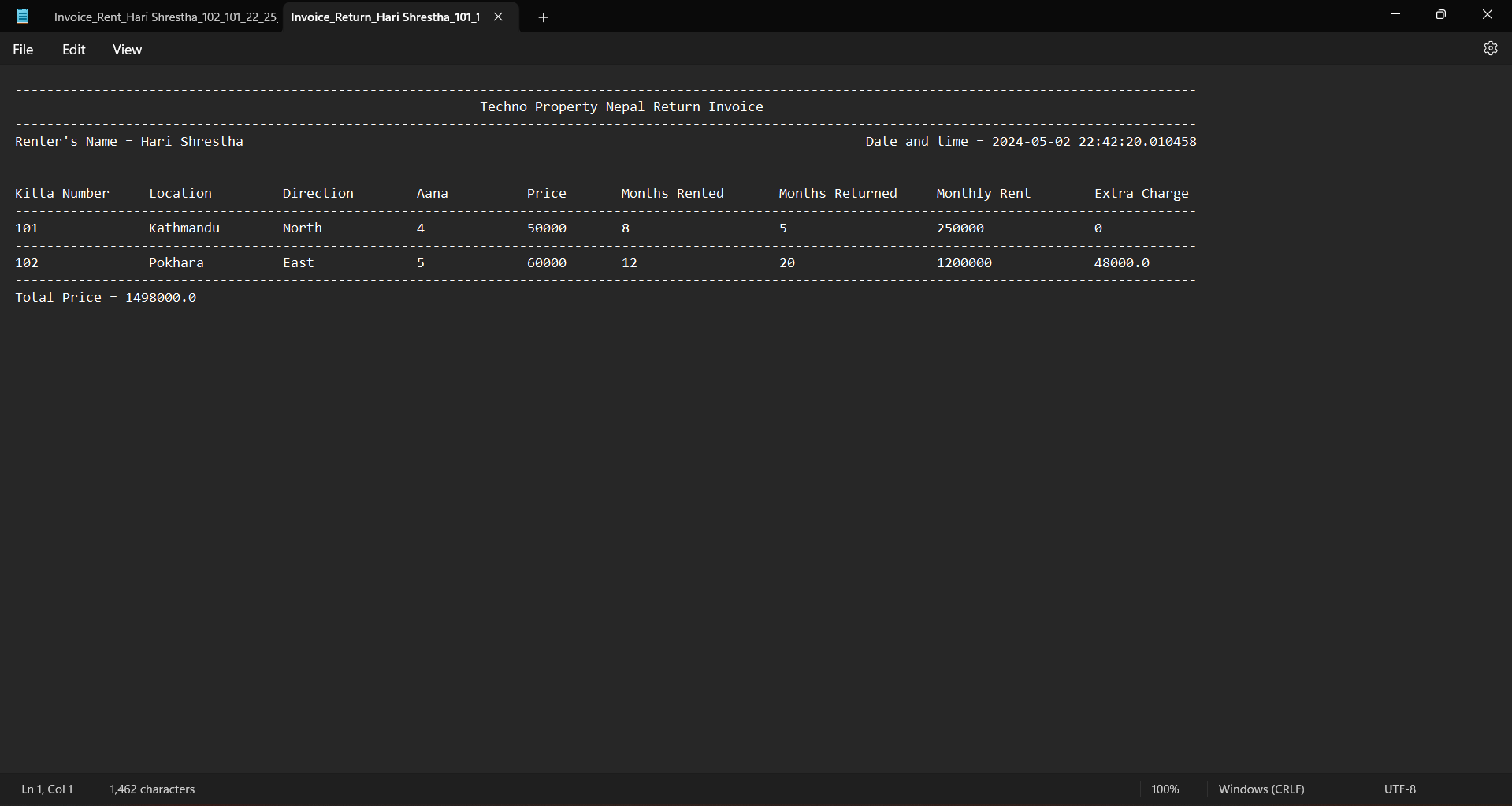


Figure 57: Test – 4 - Return invoice

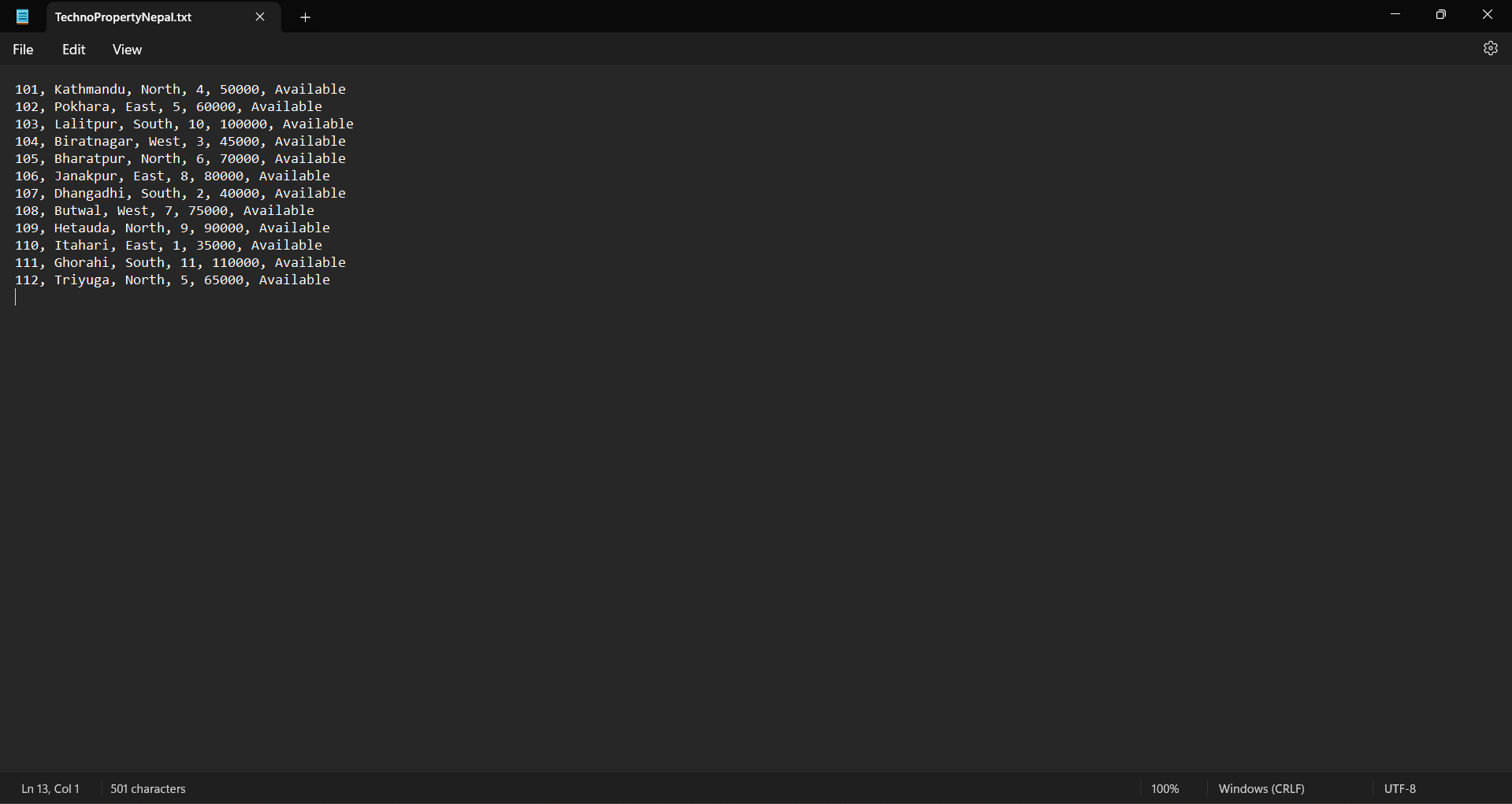


Figure 58: Test – 4 - Updated lands in land file

## 4.5 Test – 5: Showing the update in stock of land(s)

|  |  |
| --- | --- |
| Test no: | 5 |
| Objective | Showing the update in stock of land(s) |
| Action | Run the program  In the home screen, 1 is given as input  Then, in the name, Shyam Maharjan is given as input  After that, in the kitta number 108 is given as input  In the rented months, 5 is given as input and in rent again, no is given as input  Again, in the home screen 2 is given as input  Then in name Shyam Maharjan is given as input  After that, in kitta number 108 is given as input  In the rented months, 5 is given as input and in returned month 5 is given as input  In return again, no is given as input. |
| Expected Result | It will update the lands both in shell and in file. |
| Actual Result | It updated the lands both in shell and in file. |
| Conclusion | The test is successful |

Table 5: Test – 5: Showing the update in stock of land(s)

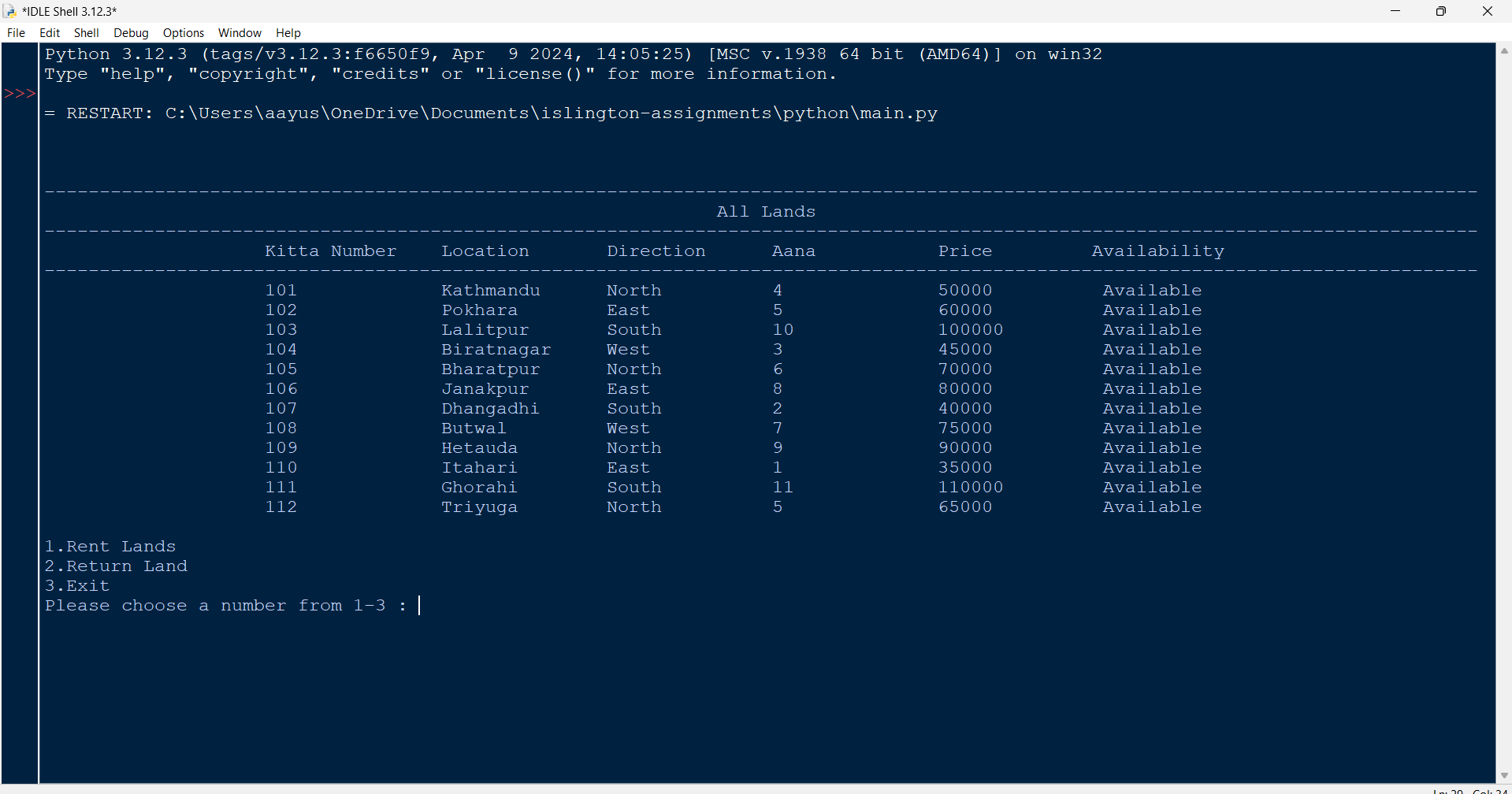


Figure 59: Test – 5 - Home screen and typing 1 as input

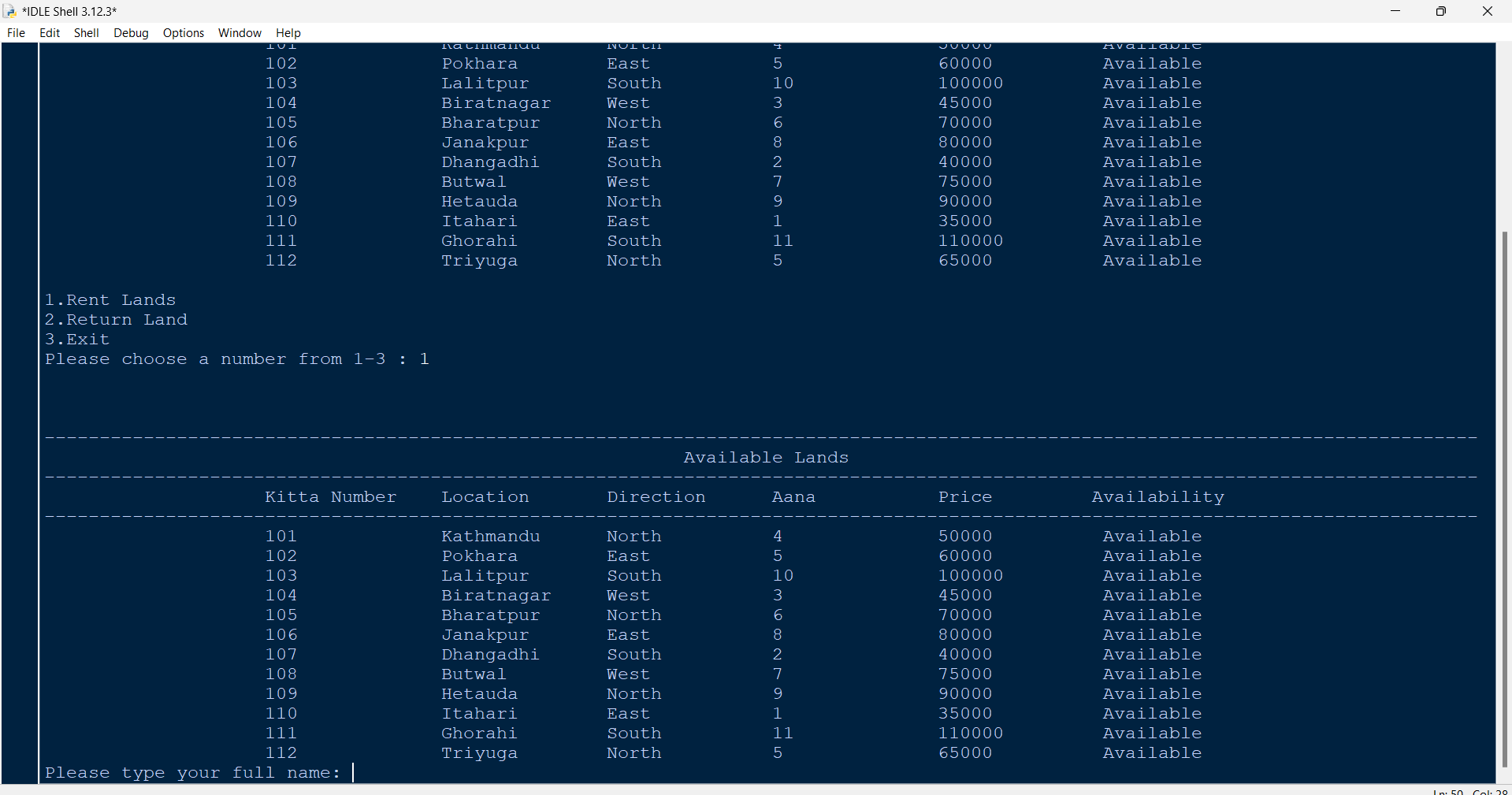


Figure 60: Test – 5 - Giving name as input

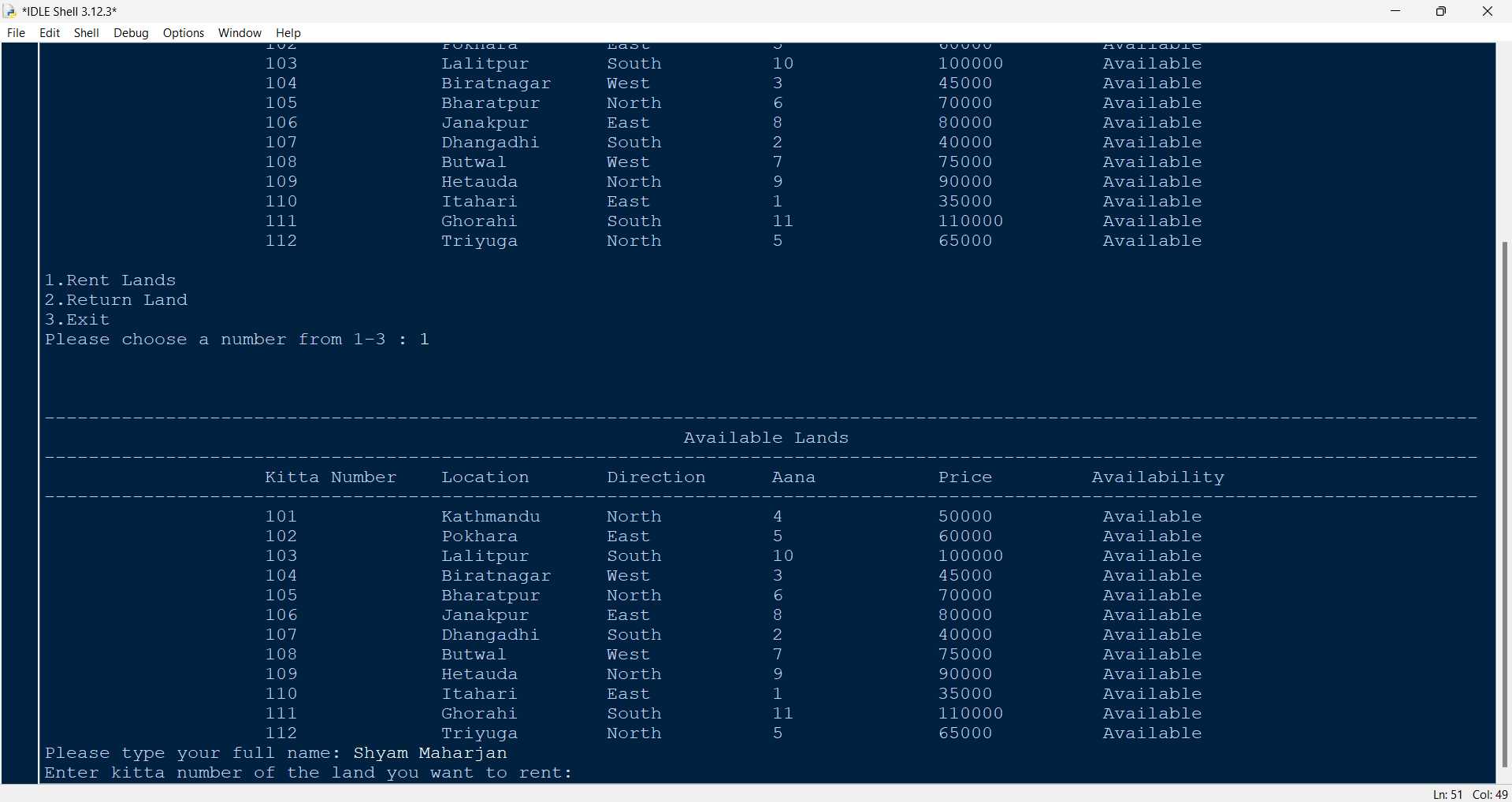


Figure 61: Test – 5 - Giving kitta number as input

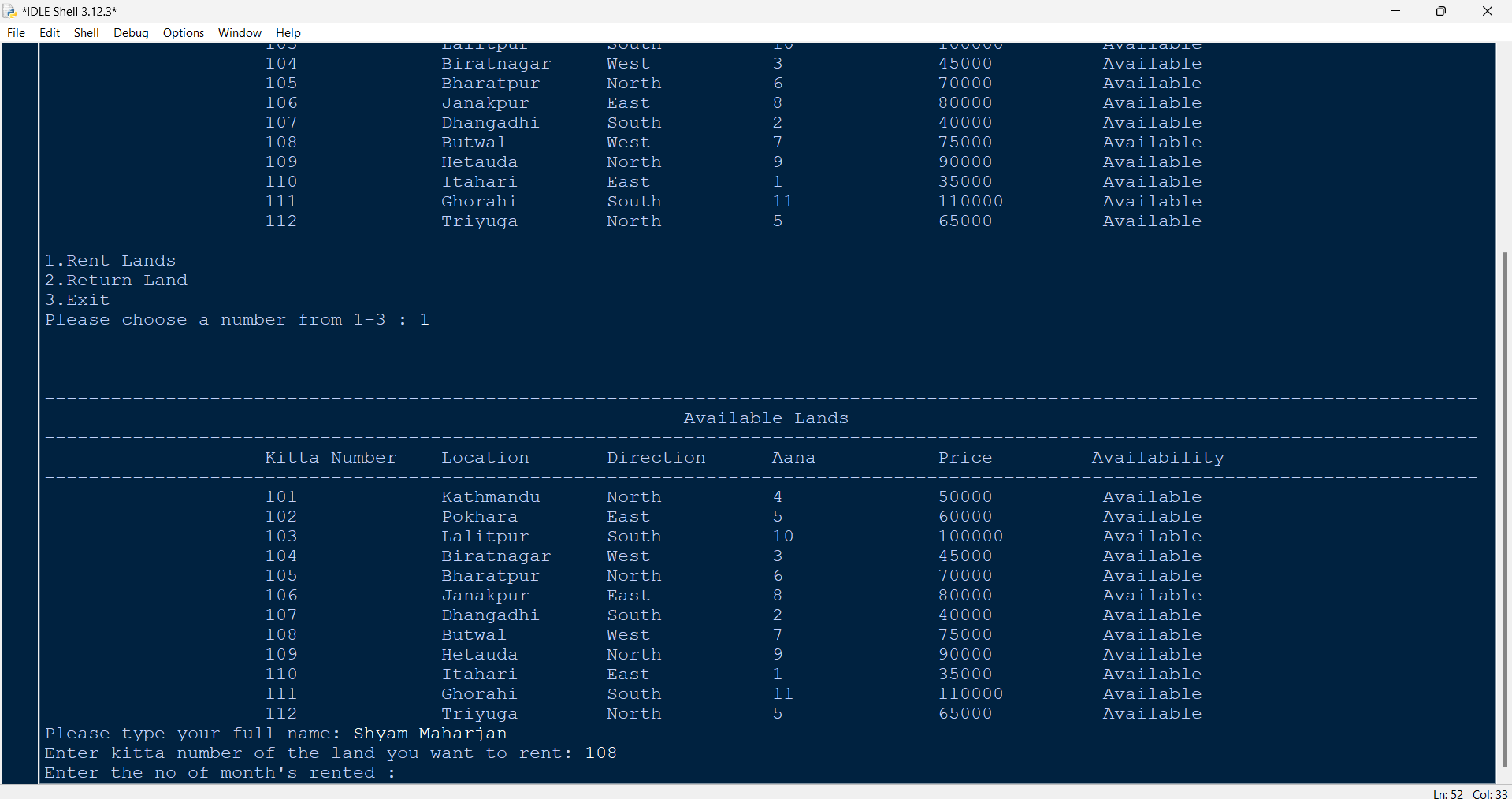


Figure 62: Test – 5 - Giving month's rented as input

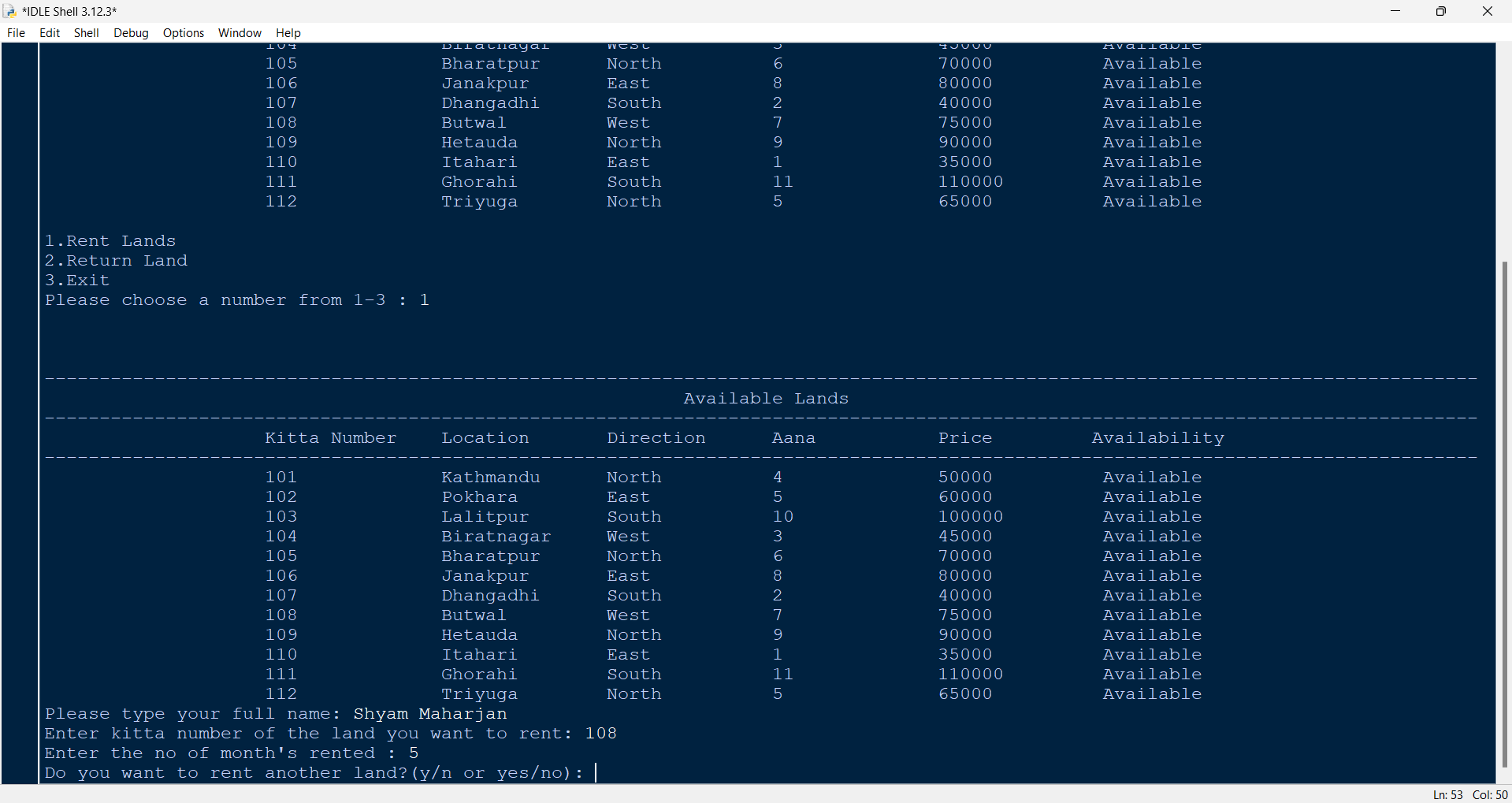


Figure 63: Test – 5 - Giving no as input in rent another land

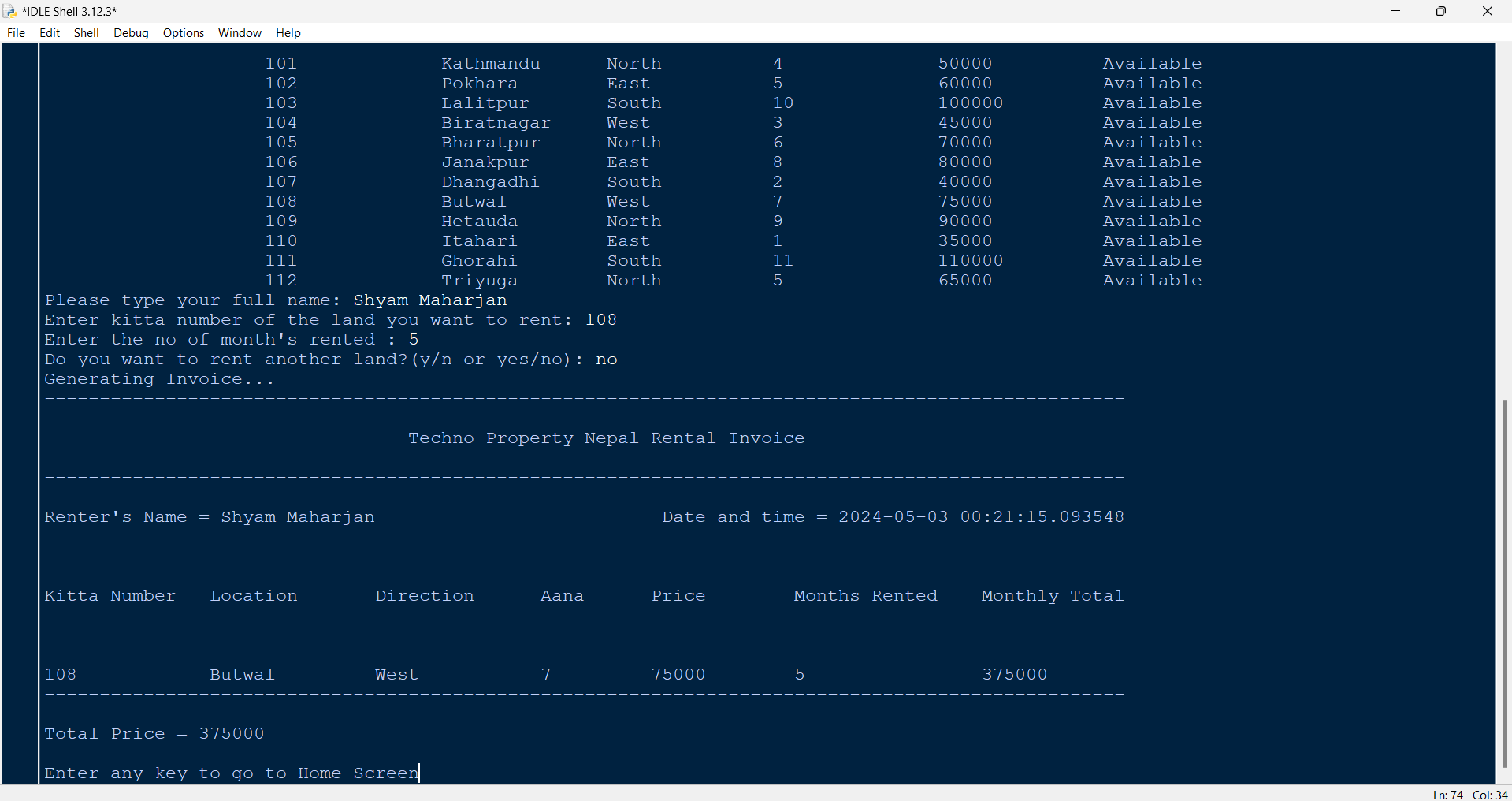


Figure 64: Test – 5 - Generating invoice and displaying the invoice

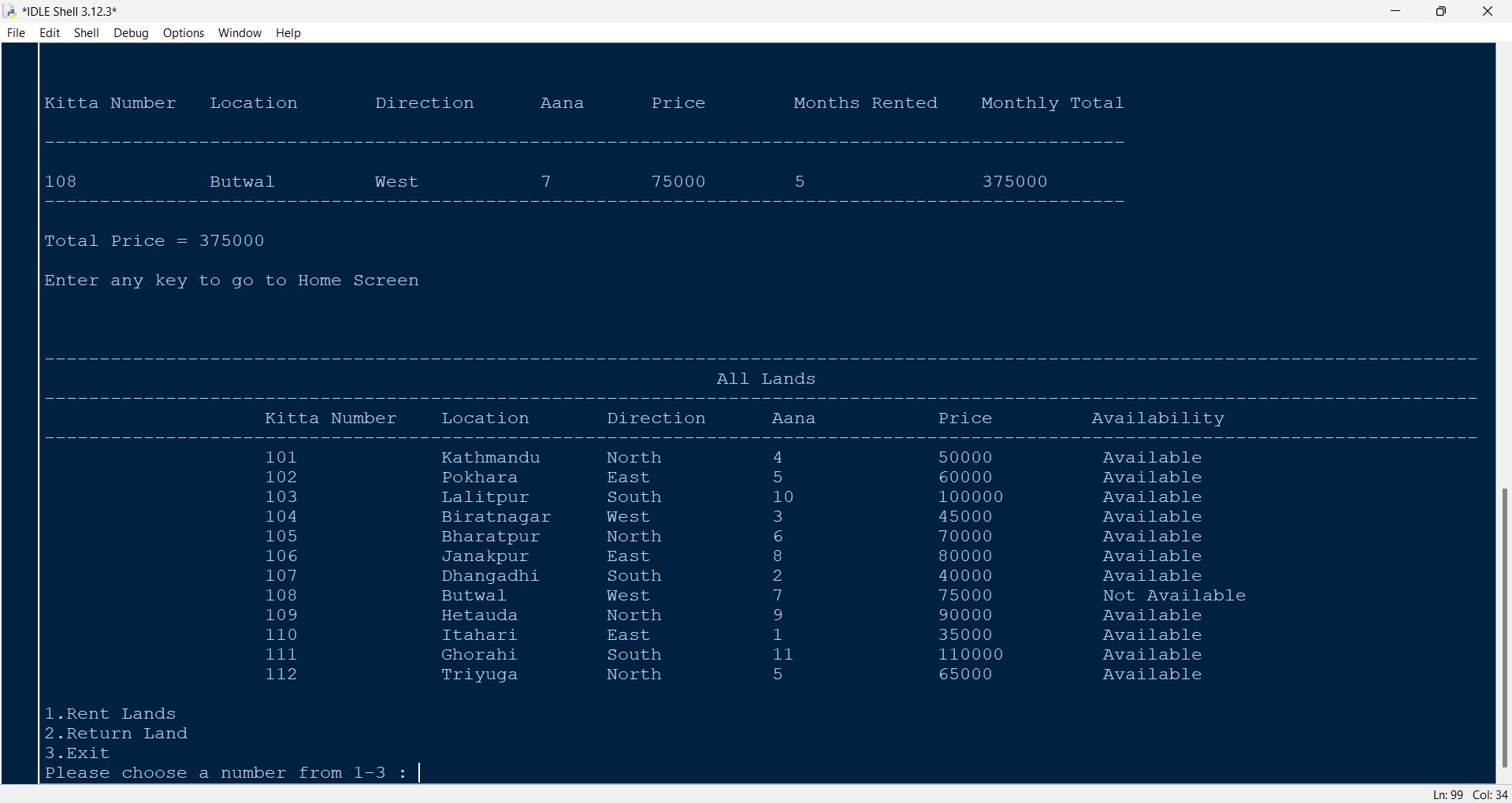


Figure 65: Test – 5 - Updated availability of the land in the home screen

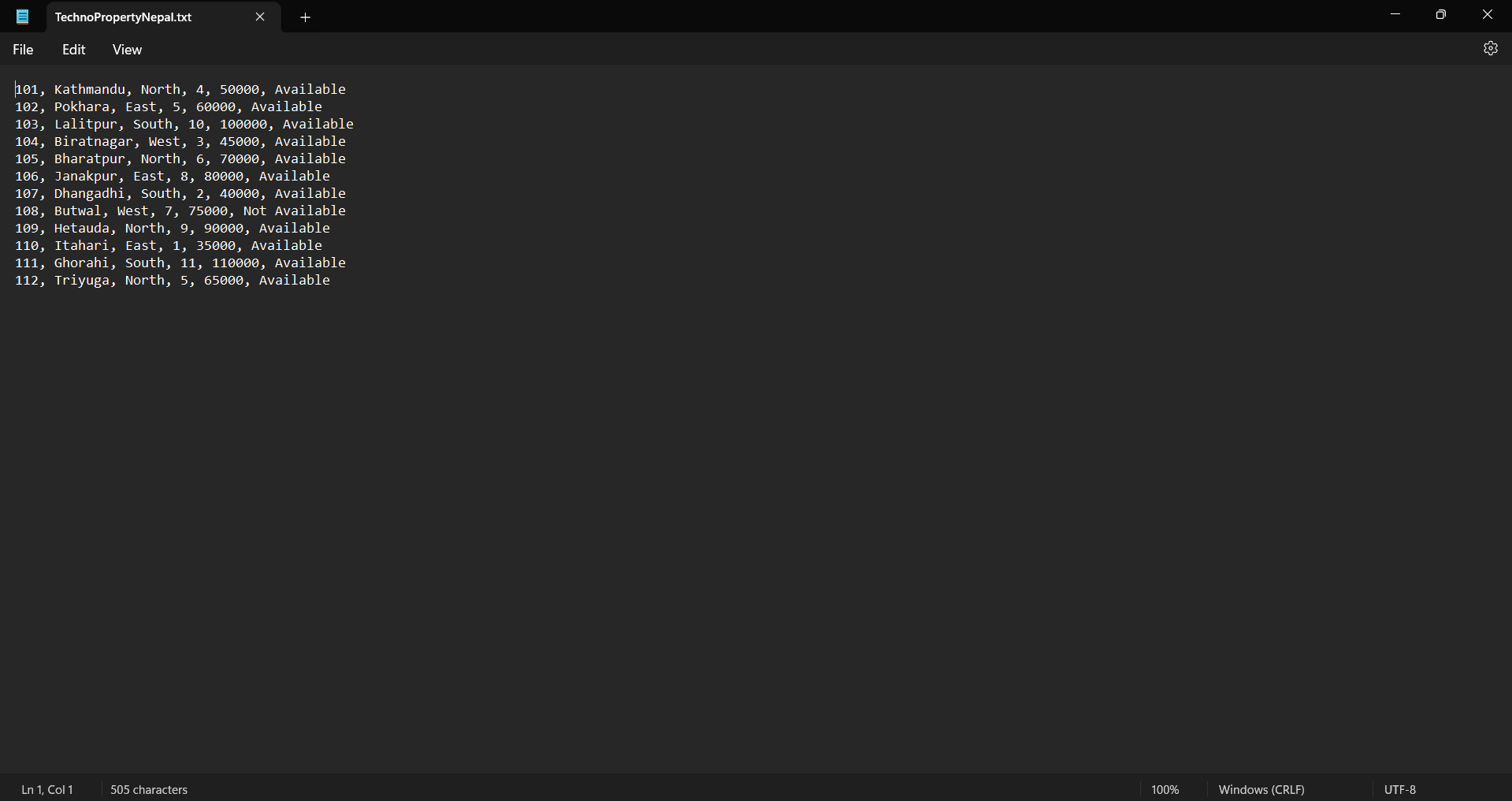


Figure 66: Test – 5 - Land file updated by the program

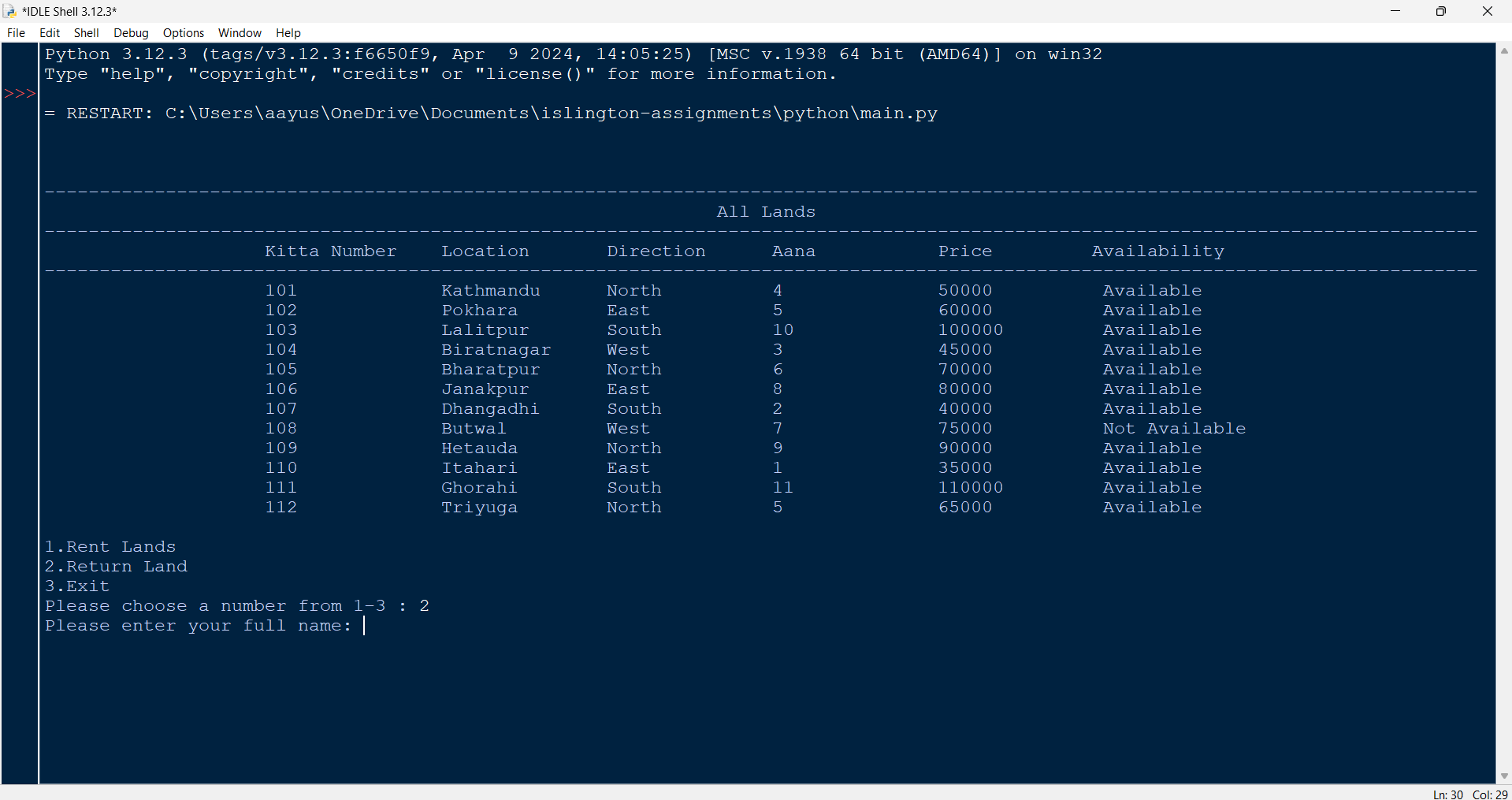


Figure 67: Test – 5 - Giving 2 in option and giving full name

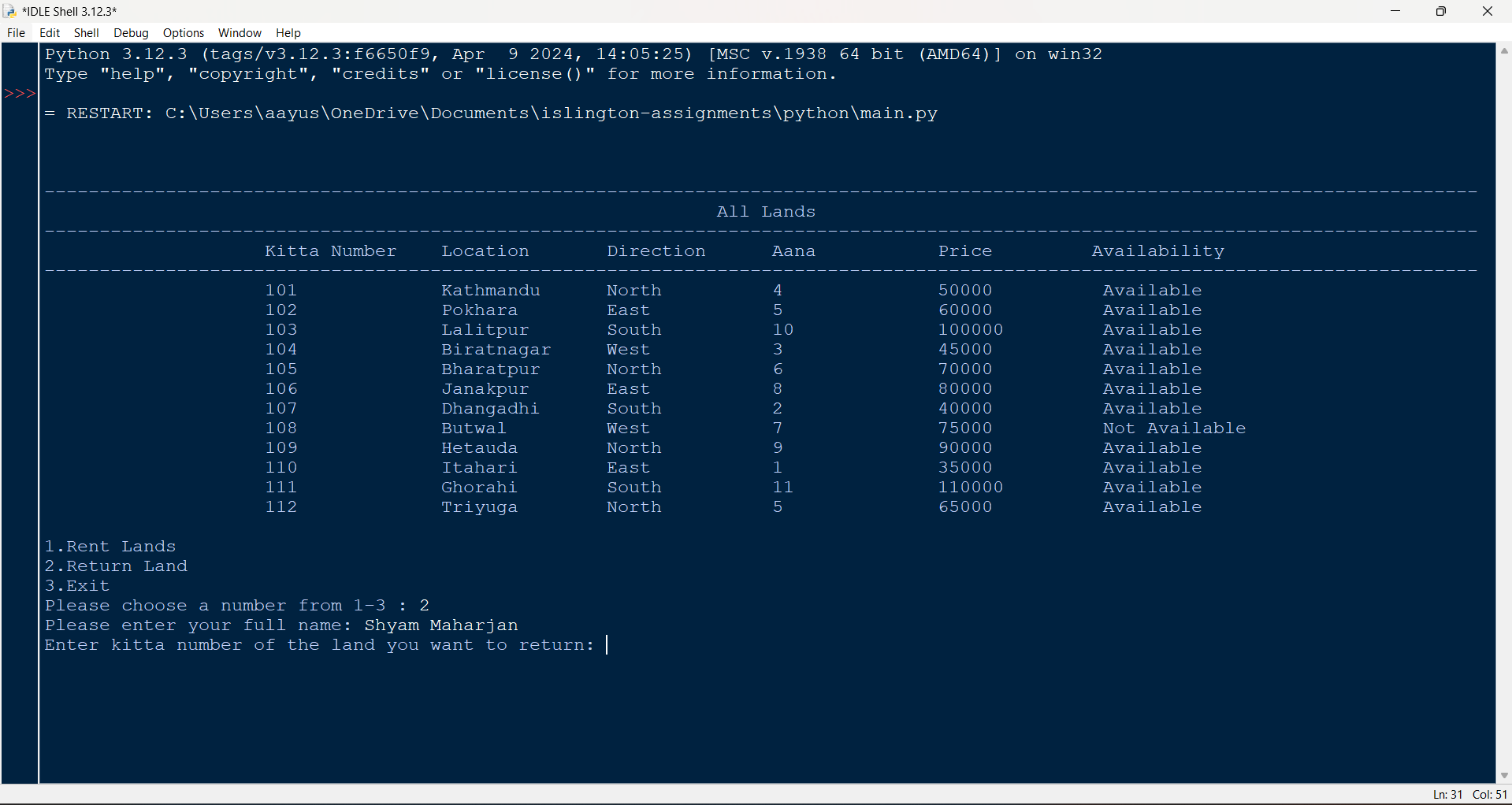


Figure 68: Test – 5 – Giving the kitta number of the land to return

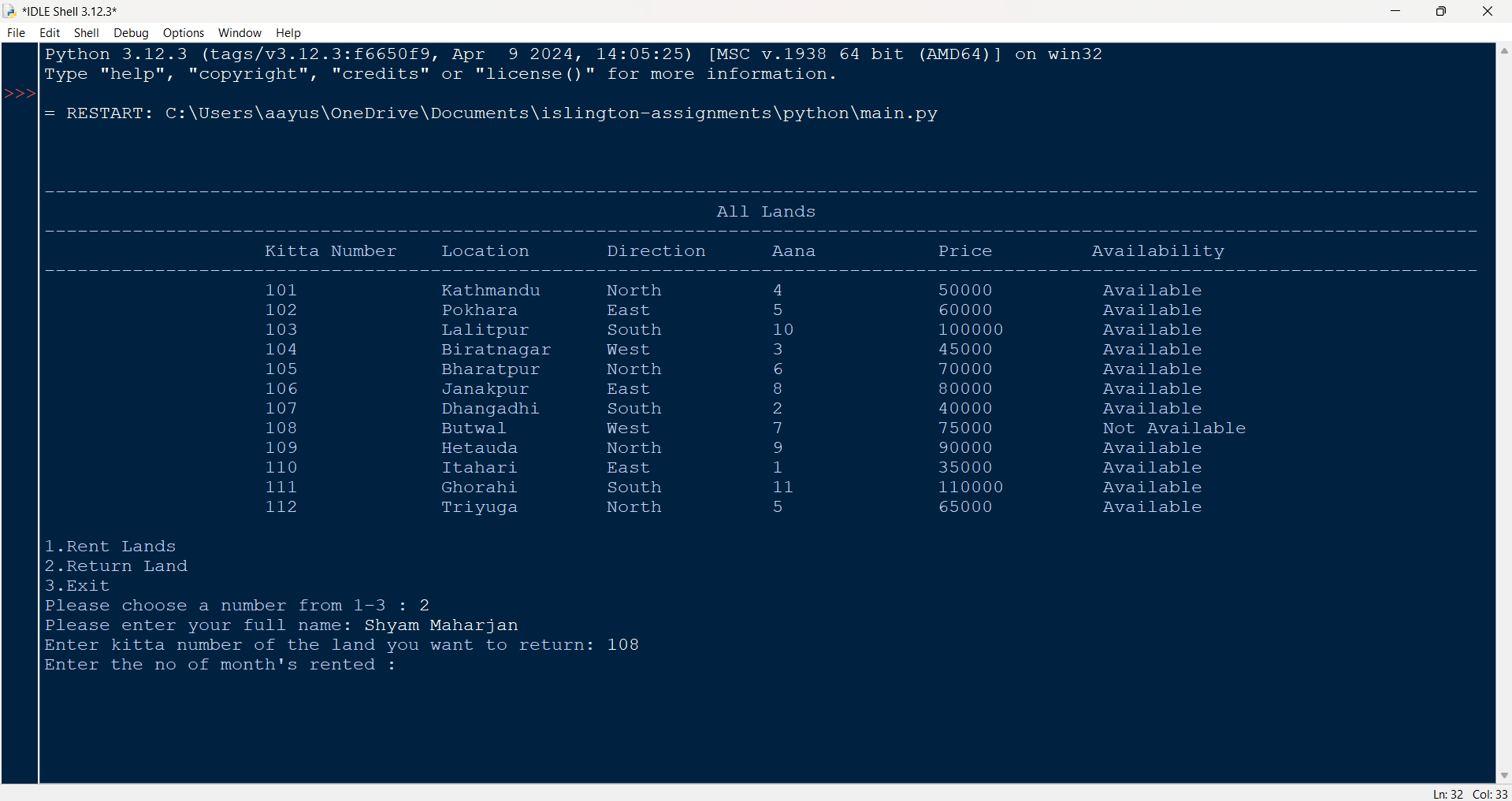


Figure 69: Test – 5 – Giving the number of months rented

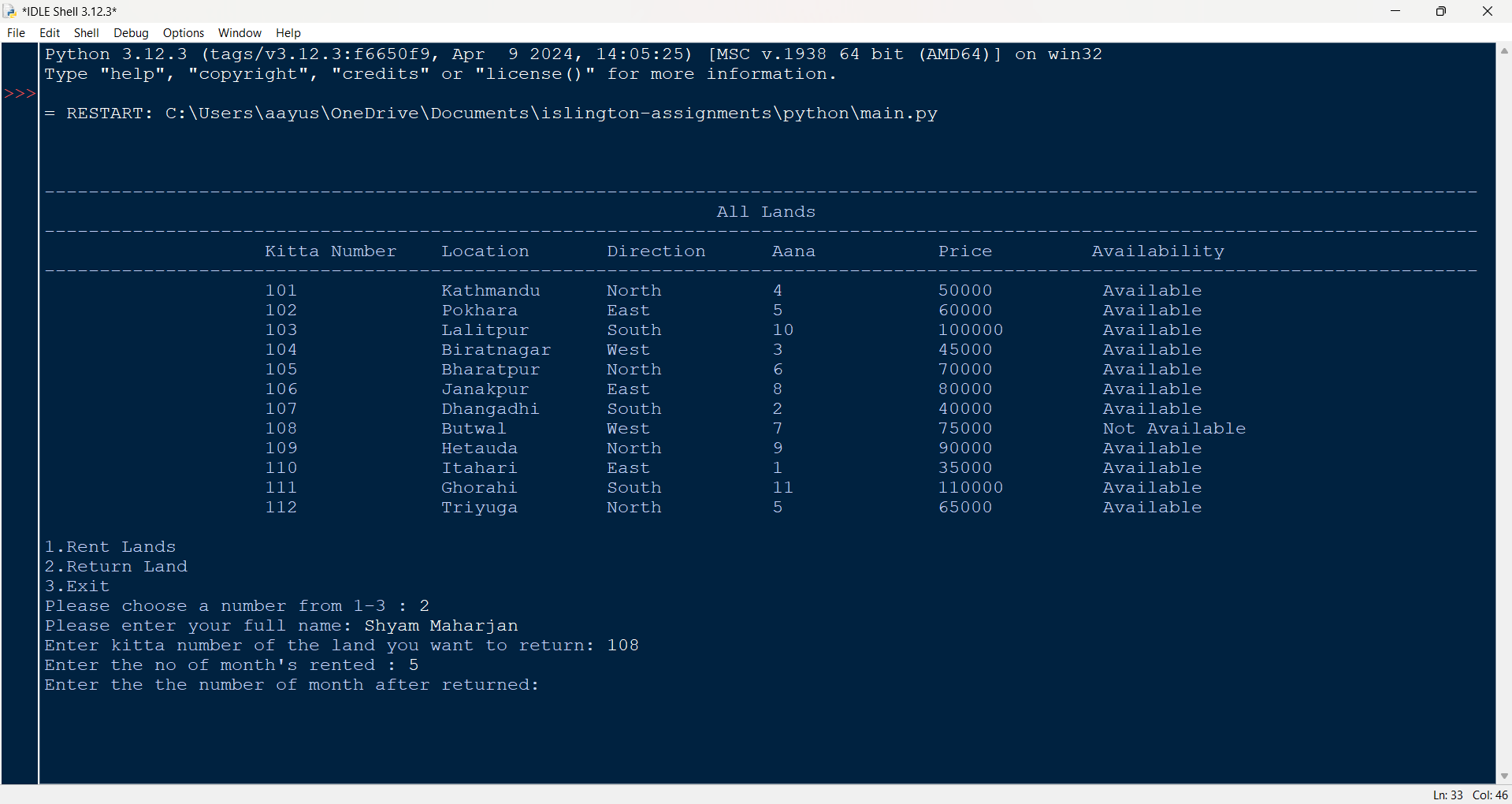


Figure 70: Test – 5 - Giving the months returned

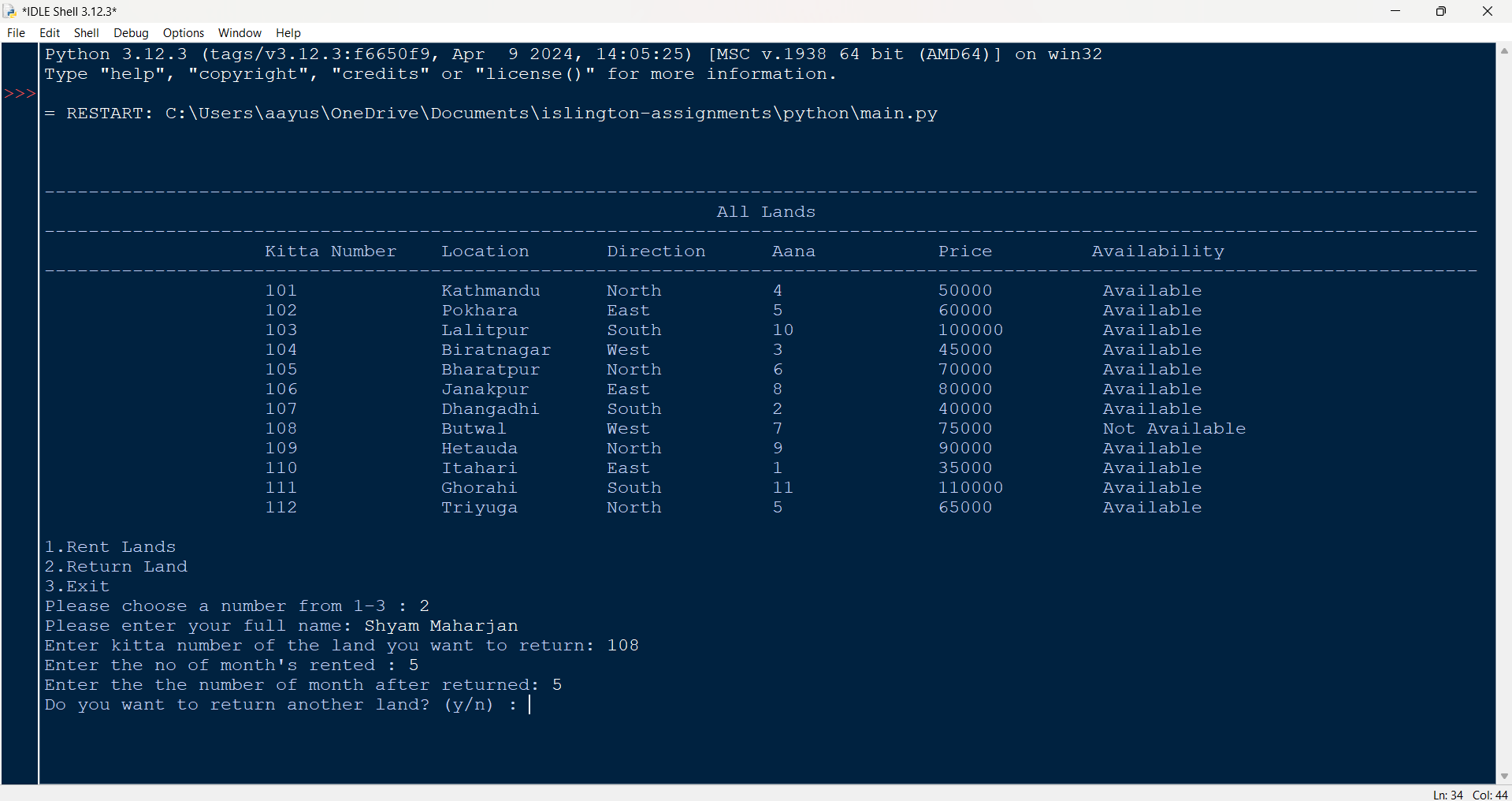


Figure 71: Test – 5 - Giving no as input in return again

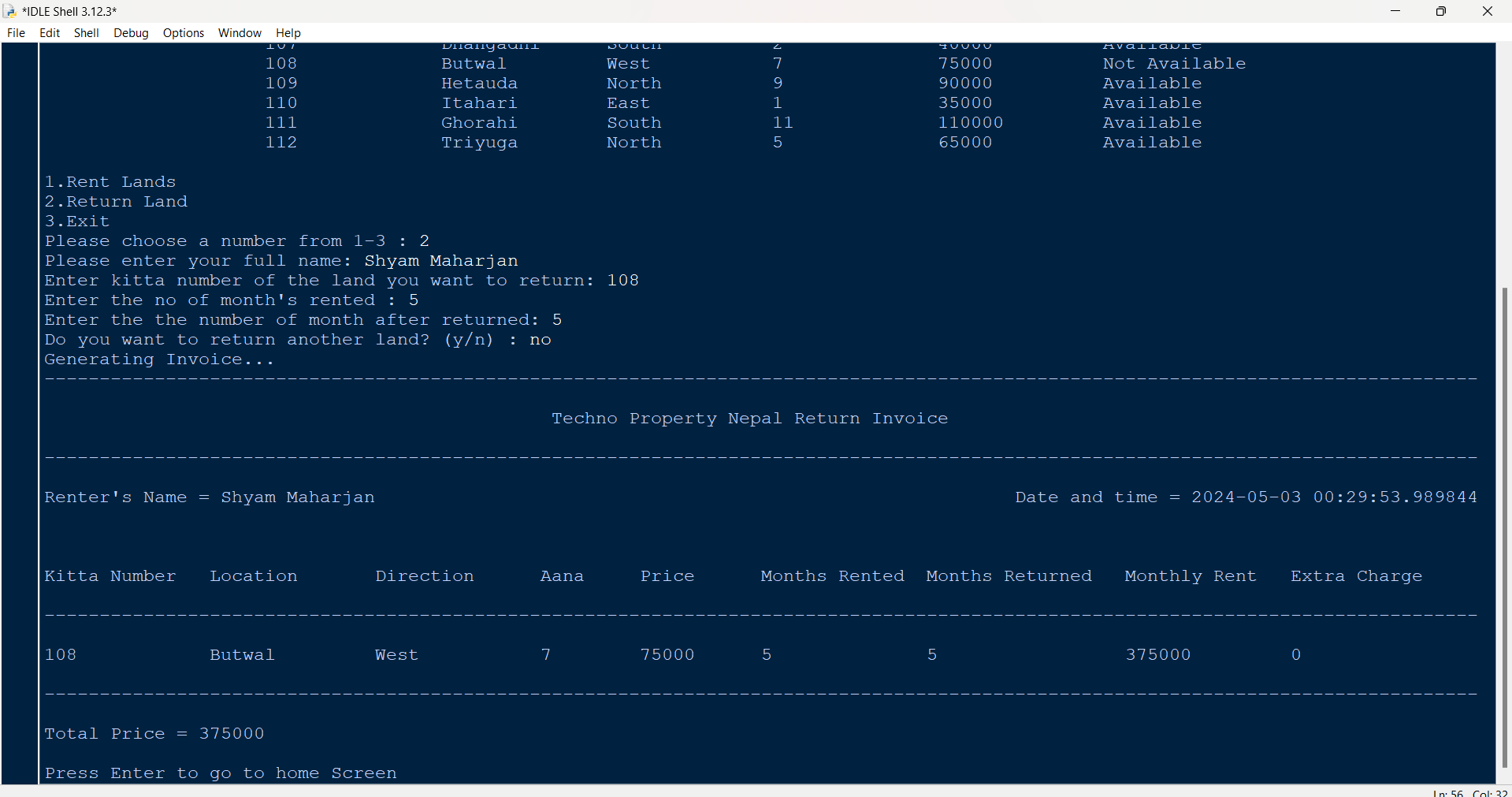


Figure 72: Test – 5 - Generated invoice and displayed on screen

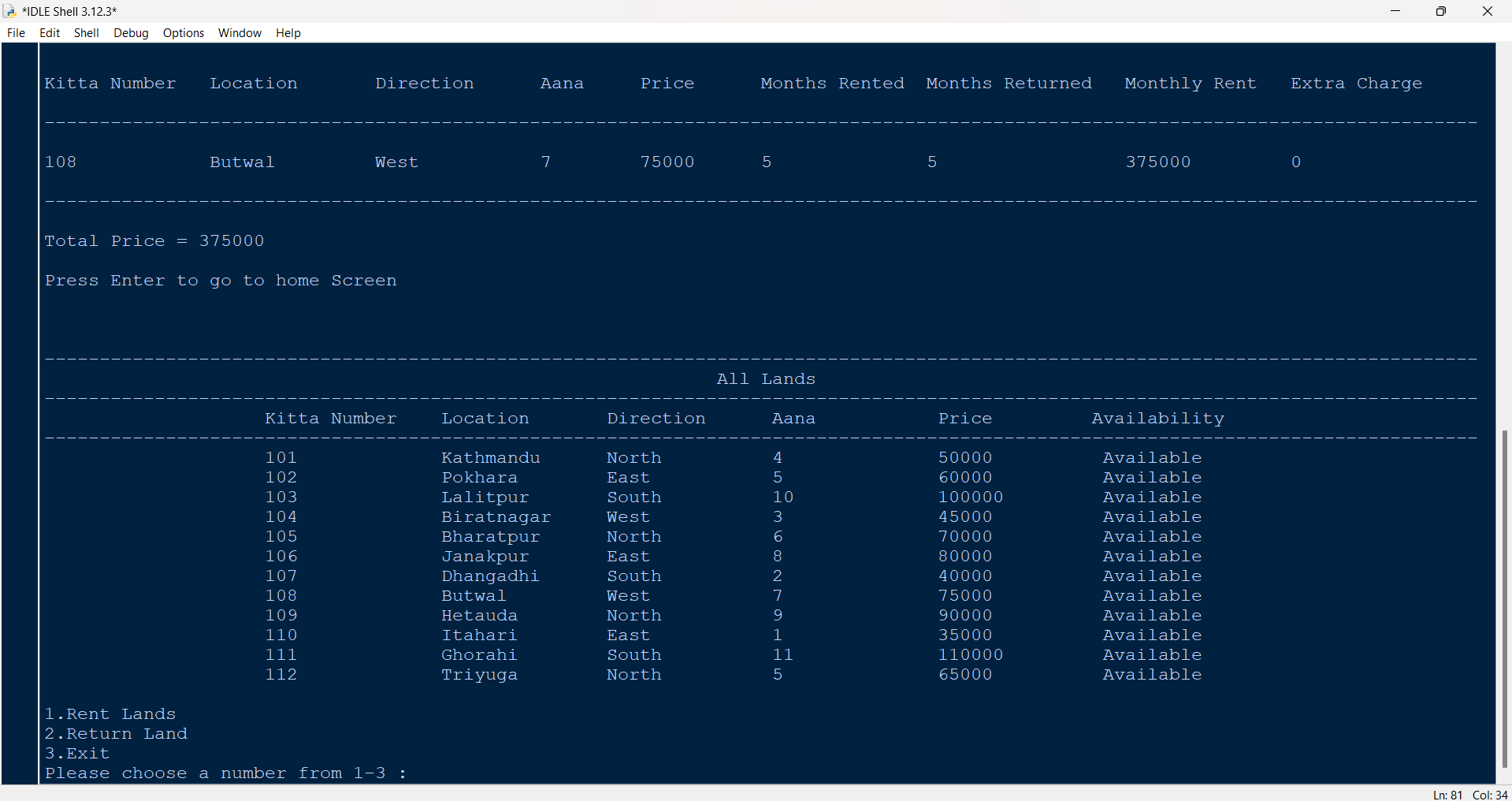


Figure 73: Test – 5 - Availability status changed to available after return

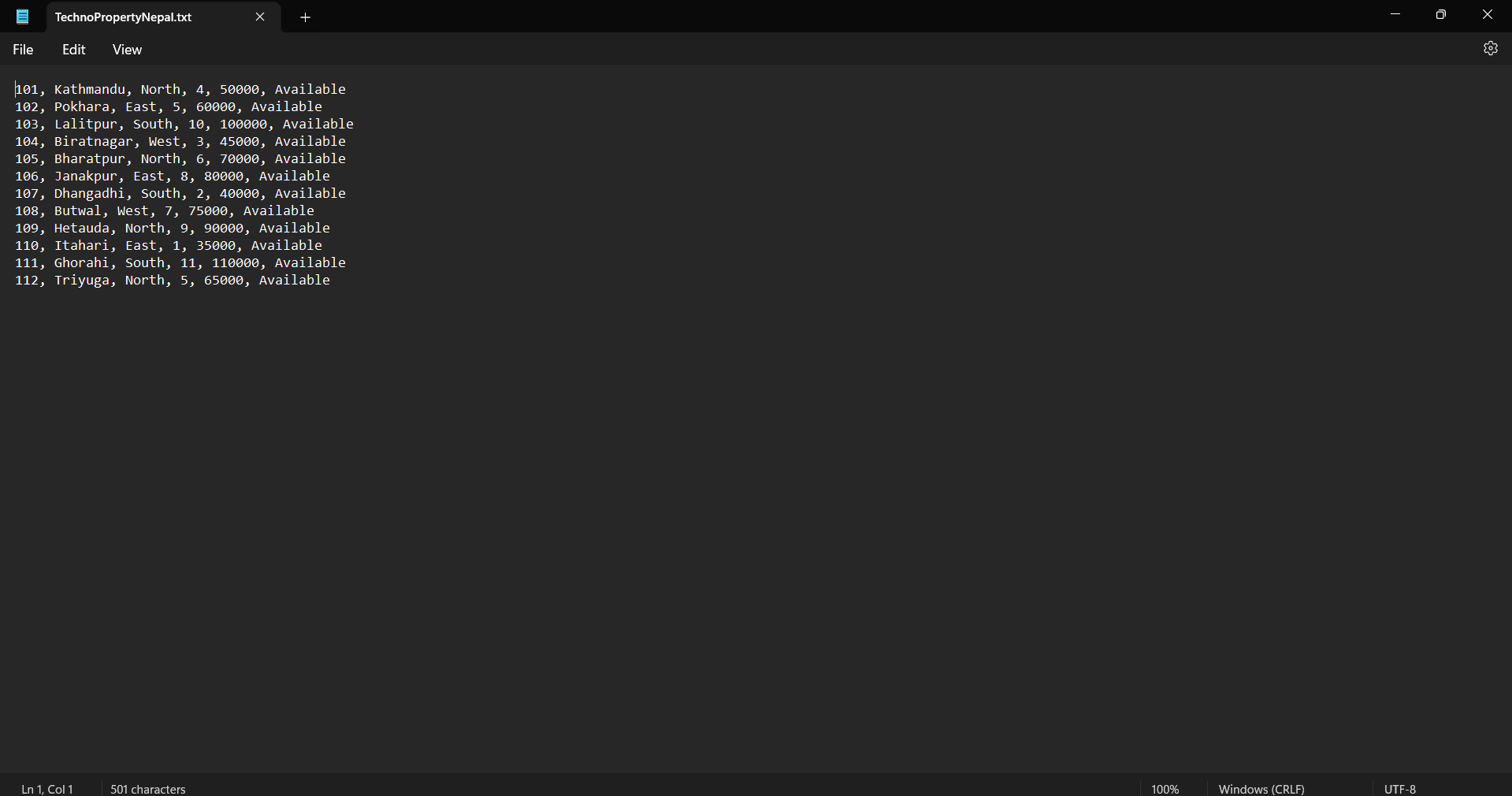


Figure 74: Test – 5 - Availability status changed in lands file

# 5. Conclusion

In this coursework, we were to develop a land rental system through which we can easily rent

In this coursework, I’ve gained a lot of knowledge about programming in python. This coursework was given to us with the objective of making us gain knowledge about programming in python and learning modular programming. While developing the program I learned how to code in a modular way using different modules such as main, operation, read and write. It also taught me how to handle files, change the details inside the file and make new unique files using python programming language. It also gave me the knowledge on how the code can be re used in multiple instances. While I have faced many problems in designing this program and implementing all the things required in the coursework, I was greatly helped by my tutorial teacher and my helpful friends who made me overcame the problems I faced in the program.

After completing this coursework, it widened my knowledge on how to code efficiently and have gained knowledge on how the modular programming works not only in python but other programming languages/ It has helped me become a better developer than before.

# 6. References

GeeksforGeeks, 2023. *Python Data Structures.* [Online]   
Available at: https://www.geeksforgeeks.org/python-data-structures/  
[Accessed 5 May 2024].

GeeksforGeeks, 2023. *What is a Flowchart and its Types?.* [Online]   
Available at: https://www.geeksforgeeks.org/what-is-a-flowchart-and-its-types/  
[Accessed 1 May 2024].

GeeksforGeeks, 2024. *Dictionaries in Python.* [Online]   
Available at: https://www.geeksforgeeks.org/python-dictionary/  
[Accessed 5 May 2024].

GeeksforGeeks, 2024. *What is python? it's uses and applications.* [Online]   
Available at: https://www.geeksforgeeks.org/what-is-python/  
[Accessed 1 May 2024].

Gillis, A. S., 2023. *What is an algorithm?.* [Online]   
Available at: https://www.techtarget.com/whatis/definition/algorithm  
[Accessed 1 May 2024].

Sheldon, R., 2023. *What is pseudocode?: Definition from TechTarget.* [Online]   
Available at: https://www.techtarget.com/whatis/definition/pseudocode  
[Accessed 2 May 2024].

Taylor, S., 2023. *What are Python Data Structures (Lists, Sets, and Tuples)?.* [Online]   
Available at: https://corporatefinanceinstitute.com/resources/data-science/python-data-structures/  
[Accessed 5 May 2024].

# 7. Appendix

## 7.1 Main.py

import read

import operation

#Variable to loop the program until the usr tells to stop

loop = True

#Looping until the loop is true

while loop:

try:

read.all\_lands()

#Prints the choice user has on the screen

print("\n1.Rent Lands \n2.Return Land \n3.Exit")

#Taking the iunput from the user

userin = int(input("Please choose a number from 1-3 : "))

#If the user input is 1,it will execute the renting of the land

if userin == 1:

#Prints all the avialable lands

read.available\_lands()

#Rents the land according to the user input

operation.rent\_lands()

#If the user input is 2, it will execute the returning of the land

elif userin == 2:

#Returns the land according to the input given by user

operation.return\_lands()

#If the user input is 3, it will exit the program

elif userin == 3:

loop = False

#Else it will ask the user to type a valid input

else:

print("Please type a valid number from 1-3")

#If the input is anyhting except a number, it will ask the user to type a valid input

except:

print("Please type a numeric value and try again")

## 7.2 Operation.py

import read

import write

def input\_month\_rented():

"""

This function takes user input for the number of months rented and

returns the month after it satisfies the condition.

The condition is that the month should be between 1 and 12.

"""

# Variable for the loop until the input is valid

month\_check = True

while month\_check:

try:

# Taking user input for the number of months rented

month = int(input("Enter the no of month's rented : "))

# If the input is greater than 12 and less than 0, print error message

if month > 12 or month <= 0:

print("Invalid Input, Months rented must be between 1-12.")

else:

# If input is valid, return month

month\_check = False

return month

except:

# If input is invalid, prints error message and ask user to try again

print("Invalid Input, Please try again")

def input\_month\_returned():

"""

This function takes user input for the number of months after returned

and returns the month if the month is positive and greater than 0

"""

# Variable to loop until the input is valid

check = True

while check:

try:

# Taking input from the user for the number of months after returned

month\_returned = int(input("Enter the number of month after returned: "))

# Checking if the input is greater than 0

if month\_returned > 0:

# If input is valid, return the month

check = False

return month\_returned

else:

# If input is negetive, print error message and ask user to try again

print("Please provide a positive number in returned months. ")

except:

# If input is invalid, print error message and ask user to try again

print("Invalid months input, Please type in numbers again. ")

def input\_kitta\_rent():

"""used to take kitta and then check for its availability and returns true if it's available"""

check = True

while check:

try:

kitta = int(input("Enter kitta number of the land you want to rent: "))

check\_kitta = read.check\_kitta(kitta)

if check\_kitta == True:

check = False

return kitta

else:

print("Kitta not available or not in the system, Please try again. ")

except:

print("Invalid Input in Kitta,Please try again")

def input\_kitta\_return():

"""

This function takes user input for the kitta number of the land

to be returned, checks its availability and returns true

if it's not available and in the system.

"""

# Variable to loop until the input is valid

check = True

while check:

try:

# Taking input from the user for the kitta number of the land

kitta = int(input("Enter kitta number of the land you want to return: "))

# Checking if the kitta number is available

check\_return\_kitta = read.return\_check\_kitta(kitta)

# If the kitta number is available, return it

if check\_return\_kitta == True:

check = False

return kitta

else:

# If the kitta number is not in system or available, print error message and ask user to try again

print("Either the kitta number has been returned or the kitta number is not in our system. Please check your kitta number and try again.")

except:

# If user input is invalid, print error message and ask user to try again

print("Invalid input in Kitta number, Please try again")

def rent\_lands():

"""

This function is used to rent the land. It takes user input for kitta number and

number of months rented. The function rents the land, updates the system and

generates an invoice.

"""

# Variable to loop until the user is done renting

rent\_land = True

# Variable to loop until the kitta number is available

kitta\_available = True

# Variable to loop until the user gives valid input if they want to rent more or not.

rent = True

# Lists to store kitta numbers and number of months rented

kittas = []

months = []

# Taking user input for their full name

name = input("Please type your full name: ")

#looping until the user rents the land

while rent\_land:

# Looping until the kitta number is available

while kitta\_available:

# Taking user input for kitta number and number of months rented

kitta = input\_kitta\_rent()

month = input\_month\_rented()

#Adding the kitta number and no of months to the list

kittas.append(kitta)

months.append(month)

# Updating the system and changing the availability of the kitta number user input to not available

write.update\_rent\_lands(kitta)

# Looping until the user doesnt gives a valid answer

rent = True

while rent == True:

# Taking user input to rent another land or not

rent\_more = input("Do you want to rent another land?(y/n): ").lower()

if rent\_more == 'yes' or rent\_more == 'y':

# If user wants to rent another land, resetting variables

kitta\_available = True

rent\_land = True

rent = False

elif rent\_more == 'no' or rent\_more == 'n':

# If user is done renting, generating invoice and ending the loops

kitta\_available = False

rent\_land = False

rent = False

print("Generating Invoice...")

write.rent\_invoice(name,kittas,months)

# Taking garbage input to go to home screen

garbage = input("Press Enter key to go to Home Screen")

else:

# If user inputs invalid input, printing error message

print("Please type a valid input and try again")

rent = True

def return\_lands():

"""

This function takes user input for the kitta number, number of months rented and number of months after returned,

updates the system, generates an invoice.

"""

# Lists to store kitta numbers, number of months rented and number of months after returned

months\_rented = []

months\_returned = []

kittas = []

# Variables to loop until the user is done returning lands

return\_land = True

return\_more = True

# Taking user input for their full name

name = input("Please enter your full name: ")

# Looping until the user is done returning lands

while return\_land == True:

# Taking user input for kitta number, number of months rented and number of months after returned

kitta = input\_kitta\_return()

month\_rented = input\_month\_rented()

month\_returned = input\_month\_returned()

# Adding the kitta number, number of months rented and number of months after returned to their corresponding list

kittas.append(kitta)

months\_rented.append(month\_rented)

months\_returned.append(month\_returned)

# Updating the system and changing the availability of the kitta number user input to available.

write.update\_returned\_kitta(kitta)

return\_more = True

# Looping until the user doesnt gives a valid answer

while return\_more:

# Taking user input to return another land or not

return\_more = input("Do you want to return another land? (y/n) : ").lower()

if return\_more == 'yes' or return\_more == 'y':

# If user wants to return another land, resetting variables

return\_land = True

return\_more = False

elif return\_more == 'no' or return\_more == 'n':

# If user is done returning, generating invoice and ending the loops

return\_land = False

return\_more = False

print("Generating Invoice...")

write.return\_invoice(name,kittas,months\_rented,months\_returned)

garbage = input("Press Enter to go to home Screen")

else:

# If user input is invalid input, printing error message

print("Please type yes or no and Please try again")

## 7.3 Write.py

from datetime import datetime

import read

def update\_rent\_lands(kitta):

"""Takes in kitta,Opens the technoproperty file to change the given kitta from available to not available"""

#Getting all the lands

lines = read.read\_lands()

#Opens the property file in writre mode

file = open('TechnoPropertyNepal.txt', 'w')

#For each line in lines

for line in lines:

#In each line it'll split the line and splits the line into parts where there is ','

parts = line.strip().split(',')

#if the kitta in parts matches the kitta given by user

if int(parts[0]) == kitta:

#Updating the line to not available

update\_line = parts[0] + ',' + parts[1] + ',' + parts[2] + ',' + parts[3] + ',' + parts[4] + ', Not Available\n'

#writing the line in the file

file.write(update\_line)

else:

#If its not the kitta number given by the user, writing the line without updating in the file

file.write(line)

file.close()

def rent\_invoice(name,kittas,months):

"""Takes name,kittas,months and combines them with time to form a unique invoice which includes all the rented land details and total price"""

#Takes in the date and time from the system

now = datetime.now()

#Changes the date and time to only hour,minute and second

current\_time = str(now.strftime("%H"+"\_"+"%M"+"\_"+"%S"))

#Set the total price to 0

total\_price = 0

#Set the total display price to 0

total\_price\_display = 0

#Generates a unique file name by combining name, kitta numbers the user buyed and the current date and time

invoice\_name = 'Invoice\_Rent\_'+name

for i in kittas:

invoice\_name = invoice\_name+"\_"+str(i)

invoice\_name = invoice\_name+"\_"+current\_time+".txt"

#Opens the invoice file with the unique invoice name in write mode

invoice\_file = open(invoice\_name,'w')

#writes the top part of the invoice with name and time in the bill

invoice\_file.write("-"\*98+"\n")

invoice\_file.write(" "\*33+"Techno Property Nepal Rental Invoice"+"\n")

invoice\_file.write("-"\*98+"\n")

invoice\_file.write("Renter's Name = "+ name+" "\*(40-len(name))+"Date and time = "+str(now)+"\n\n\n")

invoice\_file.write("Kitta Number"+" "\*3+"Location"+" "\*7+"Direction"+" "\*6+"Aana"+" "\*6+"Price"+" "\*8+"Months Rented"+" "\*4+"Monthly Total" "\n")

invoice\_file.write("-"\*98+"\n")

#Loops for each kitta number in the list

for i in range(len(kittas)):

#for each line in the lands list

for lines in read.read\_lands():

#Replaces the \n with blank space and splits word where there is ','

line = lines.replace("\n",'').split(", ")

#restts the monthly total price

monthly\_total = 0

#if kitta matches with lines kitta write the line in the file with monthly ptice

if int(line[0])== kittas[i]:

monthly\_total +=int(line[4])\*int(months[i])

invoice\_file.write(line[0]+" "\*(14-len(line[0]))+line[1]+" "\*(15-len(line[1]))+line[2]+" "\*(15-len(line[2]))+line[3]+" "\*(10-len(line[3]))+line[4]+" "\*(14-len(line[4]))+str(months[i])+" "\*(17-len(str(months[i])))+str(monthly\_total)+"\n")

invoice\_file.write("-"\*98+"\n")

total\_price +=monthly\_total

#write the total price in the file

invoice\_file.write("Total Price = " + str(total\_price)+"\n")

invoice\_file.close()

#print the top of invoice in the screen

print("-"\*98+"\n")

print(" "\*33+"Techno Property Nepal Rental Invoice"+"\n")

print("-"\*98+"\n")

print("Renter's Name = "+ name+" "\*(40-len(name))+"Date and time = "+str(now)+"\n\n\n")

print("Kitta Number"+" "\*3+"Location"+" "\*7+"Direction"+" "\*6+"Aana"+" "\*6+"Price"+" "\*8+"Months Rented"+" "\*4+"Monthly Total" "\n")

print("-"\*98+"\n")

lands = read.read\_lands()

#for each kitta in kittas list

for i in range(len(kittas)):

#for each line in lands list

for line in lands:

#Replaces the \n with blank space and splits word where there is ','

lines = line.replace("\n",'').split(",")

#reset the monthly total price

monthly\_total = 0

#Prints the land if the kitta number matches

if int(lines[0])== kittas[i]:

monthly\_total +=int(lines[4])\*int(months[i])

print(lines[0]+" "\*(14-len(lines[0]))+lines[1]+" "\*(15-len(lines[1]))+lines[2]+" "\*(15-len(lines[2]))+lines[3]+" "\*(10-len(lines[3]))+lines[4]+" "\*(14-len(lines[4]))+str(months[i])+" "\*(17-len(str(months[i])))+str(monthly\_total))

print("-"\*98+"\n")

total\_price\_display +=monthly\_total

print("Total Price = " + str(total\_price\_display)+"\n")

def update\_returned\_kitta(kitta):

"""Takes kitta as input,opens the technoproperty file and changes the availability to available of the kitta given"""

lines = read.read\_lands()

#opens the property file in write mode

file = open('TechnoPropertyNepal.txt', 'w')

#for each line in lines list

for line in lines:

#Changes the line and splits the line in parts where there is ','

parts = line.strip().split(',')

#if kitta matches the ktita given by user, change the availability to available

if int(parts[0]) == kitta:

update\_line = parts[0] + ',' + parts[1] + ',' + parts[2] + ',' + parts[3] + ',' + parts[4] + ', Available\n'

#write the line in the file

file.write(update\_line)

else:

#Writes the lie as it is if its not the land user returned

file.write(line)

file.close()

def return\_invoice(name,kittas,months\_rented,months\_returned):

"""Takes name,kittas,months\_rented,months\_returned as input and generates a unique invoice of all the lands returned and total price"""

#takes dateand time from the system

now = datetime.now()

#creates a string of date and time

current\_time = str(now.strftime("%H"+"\_"+"%M"+"\_"+"%S"))

#Stets the total price as 0

total\_price = 0

#Stets the total display price as 0

total\_price\_display = 0

#Stets the remaining month to 0

remaining\_month = 0

#Adds the name to the invoice name

invoice\_name = 'Invoice\_Return\_'+name

#Adds all the kittas given by usder into file name

for i in kittas:

invoice\_name = invoice\_name+"\_"+str(i)

#Adds the current tiem to the file name

invoice\_name = invoice\_name+"\_"+current\_time+".txt"

#opens the unique invoice file with invoice name in write mode

invoice\_file = open(invoice\_name,'w')

#writes the top part of the invoice in the file

invoice\_file.write("-"\*150+"\n")

invoice\_file.write(" "\*59+"Techno Property Nepal Return Invoice"+"\n")

invoice\_file.write("-"\*150+"\n")

invoice\_file.write("Renter's Name = "+ name+" "\*(92-len(name))+"Date and time = "+str(now)+"\n\n\n")

invoice\_file.write("Kitta Number"+" "\*5+"Location"+" "\*9+"Direction"+" "\*8+"Aana"+" "\*10+"Price"+" "\*7+"Months Rented"+" "\*7+"Months Returned"+" "\*5+"Monthly Rent"+" "\*8+"Extra Charge"+"\n")

invoice\_file.write("-"\*150+"\n")

#Loops for easch kitta given by user to return

for i in range(0,len(kittas)):

#loops for all the llands in the lands list

for lines in read.read\_lands():

#Replaces the \n with blank space and splits word where there is ','

line = lines.replace("\n",'').split(",")

#Sets the monthly price to 0

monthly\_price = 0

#Sets the extra charge to 0

extra\_charge = 0

#Charges the monthly price according to months, if the kitta number matches

if int(line[0])== kittas[i]:

monthly\_price += int(line[4])\*months\_returned[i]

if months\_rented[i] > months\_returned[i] or months\_rented[i] == months\_returned[i]:

total\_price += monthly\_price

elif months\_rented[i] < months\_returned[i]:

remaining\_month = months\_returned[i] - months\_rented[i]

extra\_charge = (0.1\*int(line[4]))\*remaining\_month

total\_price += monthly\_price + extra\_charge

#Writes the land in the invoice file with all the monthly price and extra charge

invoice\_file.write(line[0]+" "\*(16-len(line[0]))+line[1]+" "\*(17-len(line[1]))+line[2]+" "\*(17-len(line[2]))+line[3]+" "\*(14-len(line[3]))+line[4]+" "\*(13-len(line[4]))+str(months\_rented[i])+" "\*(20-len(str(months\_rented[i])))+str(months\_returned[i])+" "\*(20-len(str(months\_returned[i])))+str(monthly\_price)+" "\*(20-len(str(monthly\_price)))+str(extra\_charge)+"\n")

invoice\_file.write("-"\*150+"\n")

#Writes the total price in the return file

invoice\_file.write("Total Price = " + str(total\_price)+"\n")

invoice\_file.close()

#prints the invoice top part on the screen

print("-"\*130+"\n")

print(" "\*46+"Techno Property Nepal Return Invoice"+"\n")

print("-"\*130+"\n")

print("Renter's Name = "+ name+" "\*(72-len(name))+"Date and time = "+str(now)+"\n\n\n")

print("Kitta Number"+" "\*3+"Location"+" "\*7+"Direction"+" "\*6+"Aana"+" "\*5+"Price"+" "\*6+"Months Rented"+" "\*2+"Months Returned"+" "\*3+"Monthly Rent"+" "\*3+"Extra Charge"+"\n")

print("-"\*130+"\n")

#loops for all the lands given by user to return

for i in range(0,len(kittas)):

#loops for all the lands in the lands list

for lines in read.read\_lands():

#splits the words where there is ',' and replaces \n with blank space

line = lines.replace("\n",'').split(",")

#Sets the montholy price to 0

monthly\_price = 0

#Sets the extra charge to 0

extra\_charge = 0

#if the kitta number of the line matches with the kitta number provided by the user, it will calculate the price and display the land on the screen

if int(line[0])== kittas[i]:

monthly\_price += int(line[4])\*months\_returned[i]

if months\_rented[i] > months\_returned[i]:

total\_price\_display += monthly\_price

elif months\_rented[i] < months\_returned[i]:

remaining\_month = months\_returned[i] - months\_rented[i]

extra\_charge += (0.1\*int(line[4]))\*remaining\_month

total\_price\_display += monthly\_price + extra\_charge

elif months\_rented[i] == months\_returned[i]:

total\_price\_display += monthly\_price

print(line[0]+" "\*(14-len(line[0]))+line[1]+" "\*(15-len(line[1]))+line[2]+" "\*(15-len(line[2]))+line[3]+" "\*(9-len(line[3]))+line[4]+" "\*(12-len(line[4]))+str(months\_rented[i])+" "\*(15-len(str(months\_rented[i])))+str(months\_returned[i])+" "\*(18-len(str(months\_returned[i])))+str(monthly\_price)+" "\*(15-len(str(monthly\_price)))+str(extra\_charge)+"\n")

print("-"\*130+"\n")

print("Total Price = " + str(total\_price\_display)+"\n")

## 7.4 Read.py

def all\_lands():

"""Opens the file and prints all the lands in the file"""

#opensthe techno property file in read mode

file = open('TechnoPropertyNepal.txt','r')

print("\n\n\n"+"-"\*130)

print(' '\*61+"All Lands")

print('-'\*130)

print(" "\*20+"Kitta Number"+" "\*4+"Location"+" "\*7+"Direction"+" "\*6+"Aana"+" "\*11+"Price"+" "\*9+"Availability")

print("-"\*130)

#for each line in the file, replaces the \n with blank space '' and then splits the word where there is ',' and then prints all the lines

for line in file.readlines():

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

print(" "\*20+line[0]+" "\*(15-len(line[0]))+line[1]+" "\*(15-len(line[1]))+line[2]+" "\*(15-len(line[2]))+line[3]+" "\*(15-len(line[3]))+line[4]+" "\*(15-len(line[4]))+line[5])

file.close()

def available\_lands():

"""Opens the file and prints all the available lands only"""

#Opens the techno property file in read mode

file = open('TechnoPropertyNepal.txt','r')

print("\n\n\n"+"-"\*130)

print(' '\*58+"Available Lands")

print('-'\*130)

print(" "\*20+"Kitta Number"+" "\*4+"Location"+" "\*7+"Direction"+" "\*6+"Aana"+" "\*11+"Price"+" "\*9+"Availability")

print("-"\*130)

#for each line in the file, it replaces the \n with blank space, splits the word where there is ',' and then checks the availability of the land, if available prints the land in the terminal

for line in file.readlines():

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

if line[5].strip().lower() == "available":

print(" "\*20+line[0]+" "\*(15-len(line[0]))+line[1]+" "\*(15-len(line[1]))+line[2]+" "\*(15-len(line[2]))+line[3]+" "\*(15-len(line[3]))+line[4]+" "\*(15-len(line[4]))+line[5])

file.close()

def check\_kitta(kitta):

"""Takes kitta as input,Opens the file and checks kitta, returns true if available """

#Opens the file TechnoPropertyNepal in read mode

file = open('TechnoPropertyNepal.txt','r')

#for each line in the file,it replaces the \n with blank space then splits the words where there is ','.

for line in file.readlines():

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

#If the line is available and the user input is equals to the kitta number in the line, it returns true

if line[5].strip().lower() == "available" and int(line[0])== kitta:

return True

file.close()

def return\_check\_kitta(kitta):

"""Opens the file and checks for kitta, returns true if not available"""

#Opens the property file in read mode

file = open('TechnoPropertyNepal.txt','r')

for line in file.readlines():

#for each line in the file, it replaces the \n with blank space then splits the words where there is ','

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

#If the line is not available and the user input is equals to the kitta number in the line, it returns true

if int(line[0]) == kitta and line[5].strip().lower() == "not available":

return True

file.close()

def read\_lands():

"""Opens the TechnoPropertyNepal file in read mode and prints all the lines in the file"""

#Opens the TechnoPropertyNepal file in read mode

file = open('TechnoPropertyNepal.txt','r')

lands = []

for line in file.readlines():

line.replace("\n",'').split(",")

lands.append(line)

return lands

## 7.5 TechnoPropertyNepal.txt

101, Kathmandu, North, 4, 50000, Available

102, Pokhara, East, 5, 60000, Available

103, Lalitpur, South, 10, 100000, Available

104, Biratnagar, West, 3, 45000, Available

105, Bharatpur, North, 6, 70000, Available

106, Janakpur, East, 8, 80000, Available

107, Dhangadhi, South, 2, 40000, Available

108, Butwal, West, 7, 75000, Available

109, Hetauda, North, 9, 90000, Available

110, Itahari, East, 1, 35000, Available

111, Ghorahi, South, 11, 110000, Available

112, Triyuga, North, 5, 65000, Available