LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS4051NI Fundamentals of Computing

Assessment Weightage & Type

100% Individual Coursework

Year and Semester

2019-20 Autumn

Student Name: Prashanna GC

Group: C12

London Met ID: 19031368

College ID: NP01CP4A190249

Assignment Due Date: 8th May, 2020

Assignment Submission Date: 17th May, 2020

Table of Contents

List of Figures

List of Tables

INTRODUCTION

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. (Kuhlman, 2012)

Binary adders are digital circuits used for basic arithmetic operations in computers. An 8-bit adder that takes two 8-bit integers and operates the arithmetic operations. (Wolfram, 2020) The question of the assignment was based on the principle of using full adder to calculate the sum of two 8-bit adders.

A logic gate is a building block of a digital circuit. Logic gates have two inputs and one output and are based on Boolean algebra. One of the two binary conditions is false (high) or true (low). False represents 0, and true represents 1. Depending on the type of logic gate being used and the combination of inputs, the binary output will differ. A logic gate can be thought of like a light switch, wherein one position the output is off (0), and in another, it is on (1).

The goal of the project was to develop a software application which simulates the behaviour of a digital circuit performing integer addition. Since the computer understands only two numbers 0 and 1, and does the operation in binary module so the application required the input from the user to be converted in binary number system. And, hence it should be able to operate binary addition arithmetic operation of two 8 bit integers

The program to build had different objectives, while the input from the user the inputs needed to be between 0 and 255.Since, the exceeded 255 makes the input 9-bit. The task of building the program was mainly considered working with logic gates, binary conversion operations, bit adder, input-output operations, etc.

MODEL

The 8-bit binary adder is used to gain the arithmetic sum of two 8-bit binary numbers. It requires different logical gates to for the implementation of binary operations. Different functions are performed with the help of gates which has the major role while an 8-bit operations. The 8-bit adder produces output by adding two 8-bit binary inputs. To create an 8-bit adder, we could use eight full 1-bit adders and connecting them. The first two bits from the far right is added and the carry over is added with the next two bits and so on. Each time the carry out value from 1-bit adder is transferred as carry in over for the next 1-bit adder and the operation is carried out.

GATES

In my program, I needed to use three electronic gates to construct this full adder which are AND gate, OR gate and XOR gate.

1) AND gate

It is a digital logic gate which provides high output i.e. 1 when all of the inputs are high. IF any one of the input or all of the input is low i.e. 0 then the output is also low.

Truth Table:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

2) OR gate

It is a digital logic gate which provides high output i.e. 1 when any one of the inputs or both inputs are high. If none of the inputs are high then the output is low.

Truth Table:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

3) XOR gate

It is a digital logic gate which provides high output i.e. 1 when the numbers of high inputs are odd. If both of the inputs are low i.e. 0 or both of the inputs are high i.e. 1 then the output is low.

Truth Table:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

1-BIT FULL ADDER

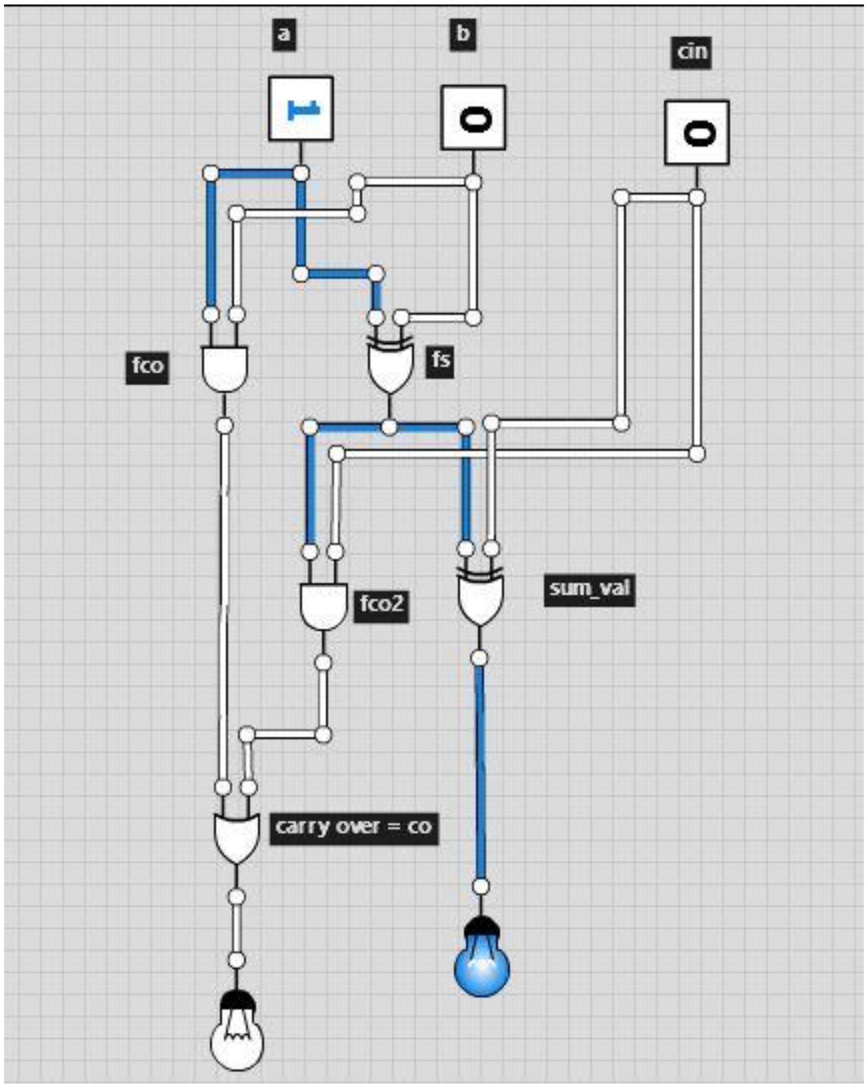


Figure 1: Representation of 1-bit full adder

The above diagram represents the 1-bit full adder logicly circuit where “cin” is the first carry in value which is initialized zero (0). In the above circuit “fs” is the first XOR value of the inputs. “fco” is the output from AND gate of the inputs. Then, after the AND gate function between “cin” and “fco” gives “fco2” whereas the XOR function between them gives the final output for the sum of the bits i.e. “sum_val”. For the carry out value, there’s an OR gate function between“fco2” and “fco”.

Truth table for Full Adder:

A	B	CARRY IN	SUM	CARRY OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

8-BIT FULL ADDER

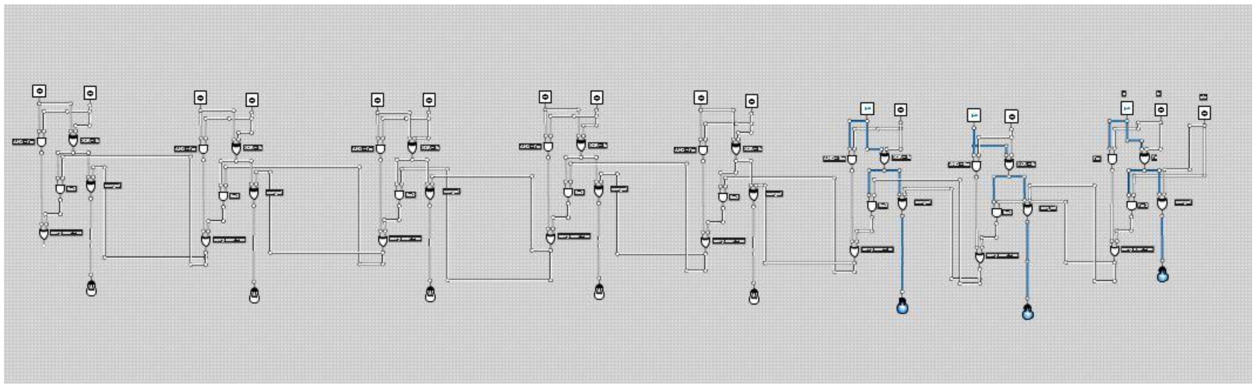


Figure 2: Representation of 8-bit full adder

The above diagram represents the 8 bit full adder logicly circuit where “cin” is the first carry in value which is initialized zero (0). In order to create a Full 8-bit adder, I used eight Full 1-bit adders and connect them. This way, the least significant bit on the far right will be produced by adding the first two bits, and then it will carry out a bit to the next two bits to add. This will continue seven more times until it is done.

Since the 8 bit full adder is created by connecting eight 1- bit full adders so the input in the circuit are same. “fs” is the first XOR value of the inputs. “fco” is the output from AND gate of the inputs. Then, after the AND gate function between “cin” and “fco” gives “fco2” whereas the XOR function between them gives the final output for the sum of the bits i.e. “sum_val”. For the carry out value, there’s an OR gate function between“fco2” and “fco”. The carry value keeps changing as per the carry of each bit. The carry out from the first two bits becomes the carry in value for the next two bits and adds with it. The Full adder above adds two bits and the output is at the end. So if we do this eight times, we would have an 8-bit adder.

I chose to do a design like this because it is simple and easy to read. The advantages of this design are being able to fix an error on all the Full Adders by just changing one because they are all copies of the same thing. The inputs are easy to distinguish from one another and the outputs are easily read on the bottom. Here we have a hierarchy of inputs on the top, then the adders in the middle, then finally the outputs on the bottom.

In the above 8 bit full adder circuit, I have taken example of two numbers i.e. 5 and 2 whose binary conversion is “00000101” and “00000010” respectively. The binary addition of these two numbers is “00000111”. We can know the binary addition from figure no. 2 with the help of the bulbs. The glowing bulb represents “1” and the other represents “0”.

Truth table for “5” and “2” based on 8-Bit Full Adder

Binary conversion of 5	Binary conversion of 2	CARRY IN	SUM	CARRY OUT
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	0	0	1	0

8-BIT PARALLEL CIRCUIT

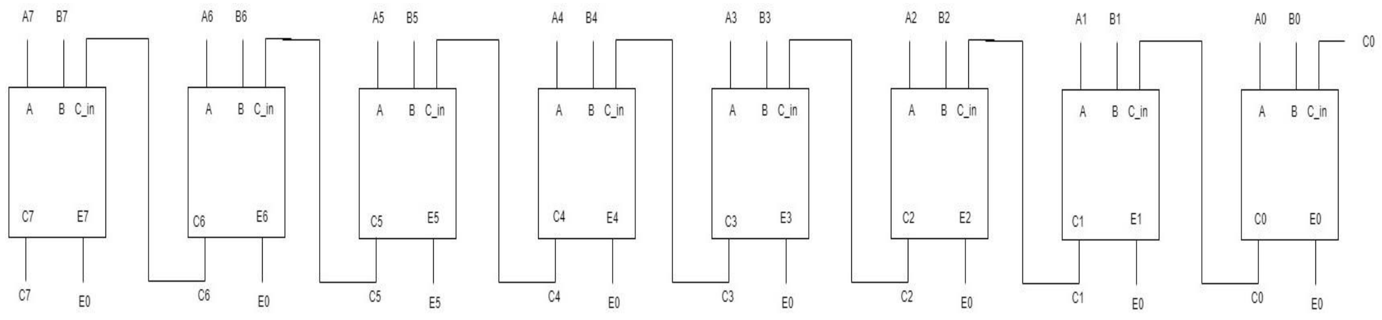


Figure 3: 8-bit block diagram

A binary parallel adder is a digital function that produces arithmetic sum of two binary numbers in parallel. It consists of full-adder combinational arrangement thus, the output carry from one full adder connected to the input carry of next full- adder. In 8 bit binary parallel adder there are 8 full adder connected in a parallel way. In this circuit the addition is done through the circuit.

In the above block diagram for 8 bit adder “A” and “B” represents the input values and “C” is the carry in and out value and “E” represents the sum as an output. The direction for carry in and out is shown with the lines in the diagram. At the beginning the carry in i.e. “C0” is initialized zero (0) in the program. Then the respective carry out from the addition of the two bits becomes the carry in value for the next binary addition.

ALGORITHM

An algorithm is a sequence of steps to solve a particular problem or algorithm is an ordered set of unambiguous steps that produces a result and terminates in a finite time Algorithm has the following characteristics:

i) Input: An algorithm may or may not require input.
ii) Output: Each algorithm is expected to produce at least one result.
iii) Definiteness: Each instruction must be clear and unambiguous.
iv) Finiteness: If the instructions of an algorithm are executed, the algorithm should terminate after finite number of steps

Algorithm of my program.

```
Step 1: START
Step 2: PRINT Welcome to the application
Step 3: INPUT first number say Input1
Step 4: IF Input1 > 255 or Input1 < 0
Step 5: PRINT Invalid
Step 6: REPEAT step 2
Step 7: IF Input1 < 255 or Input1 > 0
Step 8: CONTINUE
Step 9: INPUT second number say Input2
Step 10: IF Input2 > 255 or Input2 < 0
Step 11: PRINT Invalid
Step 12: REPEAT step 9
Step 13: IF Input2 < 255 or Input2 > 0
Step 14: CONTINUE
Step 15: Binary addition= Binary_Addition.Binary Addition (Input1, Input2)
Step 16: DISPLAY Binary addition
Step 17: INPUT Do you wish to continue.
Step 18: IF INPUT==”no”
Step 19: BREAK
Step 20: PRINT Thanks for using the application
Step 21: END
```

PSEUDOCODE

1) Pseudocode for greeting module

```
START
          BUILD def Greeting():
          DO
                    DISPLAY Welcome to the application
          ENDDO
          BUILD def GreetingAtEnd():
          DO
                    DISPLAY Thanks for using the application
          ENDDO
END
```

2) Pseudocode for Gates module

```
START
          BUILD def and_gate(x,y):
          DO
                    IF x==y and y==1
                              RETURN 1
                    ELSE
                              RETURN 0
          ENDDO
          BUILD def OR_gate(x,y):
```

```
    DO
        IF x==0 and y==0
            RETURN 0
        ELSE
            RETURN 1
    ENDDO
    BUILD def XOR_gate(x,y):
    DO
        IF x==0 and y==0 or x==1 and y==1
            RETURN 0
        ELSE
            RETURN 1
    ENDO
END
```

3) Pseudocode for reverse module

```
START
    BUILD def reverse(bit):
    DO
        GIVE actualBinary=[]
        GIVE actualBinaryNum=””
        FOR i in range (len(bit)-1,-1,-1)
            INITIALIZE actualBinary.append(bit[1])
```

```
                INITIALIZE actualBinaryNum= actualBinary + str(bit[1])
        RETURN
    ENDDO
END
```

4) Pseudocode for Binary Conversion module

```
START
    BUILD def conversion(Input1)
    DO
        GIVE bit=[]
        GIVE counter1 = 0
        WHILE counter1!=8
            INITIALIZE remainder=Input1%2
            INITIALIZE bit.append(remainder)
            INITIALIZE Input1=Input1//2
            INITIALIZE counter+=1
        RETURN bit
    ENDDO
END
```

5) Pseudocode for bit_adder module

```
START
```

```
    BUILD def binary_addition(num1,num2):
    DO
        GIVE cin=0
        GIVE List = []
        FOR i in range (len(num1)-1,-1,-1)
            INITIALIZE a=int(num[i])
            INITIALIZE b=int(num2[i])
            INITIALIZE fs =g.xor_gate(a,b)
            INITIALIZE sum_val=g.xor_gate(fs,cin)
            INITIALIZE fco=g.and_gate(a,b)
            INITIALIZE fco2=g.and_gate(fs,cin)
            INITIALIZE co=g.or_gate(fco,fco2)
            INITIALIZE cin = co
            INITIALIZE List.append(sum_val)
            INITIALIZE List2=reverse.reverse(List)
        RETURN List2
    ENDDO
END
```

6) Pseudocode for main module

```
START
    DO
        DECLARE continueLoop=True
        WHILE  continueLoop==True
            DECLARE continueNumber1=False
            WHILE continueNumber1==False
                TRY
                    INITIALIZE INPUT Input1
                    IF (Input1<0 or Input1>255)
                        DISPLAY Invalid !!!
                        CONTINUE
                    ELSE
                        DECLARE continueNumber1=True
                EXCEPT
                        DISPLAY Please enter valid number
    ENDDO
    DO
            DECLARE continueNumber2=False
            WHILE continueNumber2==False
                TRY
                    INITIALIZE INPUT Input2
                    IF (Input2<0 or Input1>255)
                        DISPLAY Invalid !!!
                        CONTINUE
                    ELSE
                        DECLARE continueNumber2=True
                EXCEPT
                        DISPLAY Please enter valid number
    ENDDO
    DO
        INITIALIZE conversion1=Binary_Conversion.conversion(Input1)
        INITIALIZE conversion2=Binary_Conversion.conversion(Input2)
        INITIALIZE binaryNumber1=reverse.reverse(conversion1)
        INITIALIZE binaryNumber2=reverse.reverse(conversion2)
        INITIALIZE binary_addition= bit_adder.binary_addition
        (binaryNumber1,binaryNumber2)
DISPLAY ("Binary addition of",Input1,"and",Input2,"is: ",binary_addition)
ENDDO
    DO
        INITIALIZE INPUT continuous=input("Do you wish to continue?").lower()
        IF continuous=="no"
            BREAK
    ENDDO
END
```

FLOWCHART

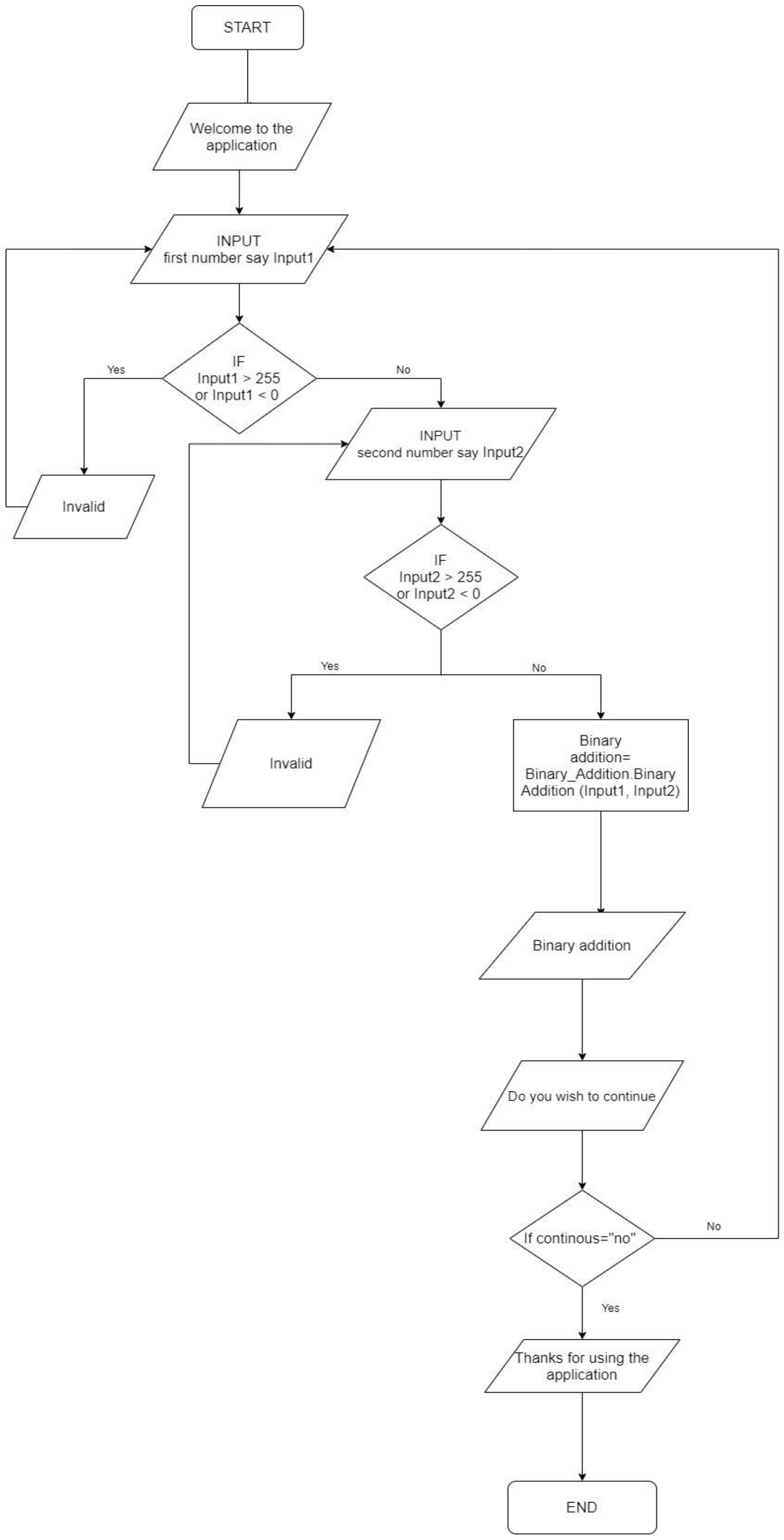

START
Welcome to the application
INPUT first number say Input1
IF Input1 > 255 or Input1 < 0
Yes
No
Invalid
INPUT second number say Input2
IF Input2 > 255 or Input2 < 0
Yes
No
Invalid
Binary addition= Binary_Addition.Binary Addition (Input1, Input2)
Binary addition
Do you wish to continue
If continous="no"
No
Yes
Thanks for using the application
END

Figure 4: Flowchart representation of the program

DATA STRUCTURE

1) List

In python, list constructor returns a list. A list is mutable which refers that its content can be changed without changing its identity. Lists are used to store various items. To build a list [ ] square brackets are used and the elements in a list is separated by a comma. If no any element is provided in a list then it’s an empty list. List manipulations can also be done. Adding a new element, removing a element, etc. are some list manipulations. (Anon., 2020)

In my program, a list named “List” is built in binary_addition function. It stores the value of the sum of each bit of the binary conversion which is again reverse and the final output is store in the other list named “List2”. The other list named “bit” where the remainder of the decimal number after being divided by 2 is stored and this list is return in the conversion function. The other list named “actualBinary” is defined in reverse function which stores the reversed value of the binary conversion and returned.

2) Int

Int function is a function which is used to convert any specified value into integer number. Integer represents the numbers which are not in decimal, fraction or in square root. Even if the object is in string or simple not a number then int can be used to convert it into an integer object.

In my program, two input functions are used which provides the place for the user to enter two numbers to show their binary addition. And, these input functions are converted in int such that the inputs provided by the user are to be in integer value so, if float value and string value is provided then an error message in displayed. Although, Ints can be either positive or negative value but this program is built in a way to function for only positive values.

3) String

In short, strings are immutable sequence of characters. There are a lot of methods to ease manipulation and creation of strings. Using a double quotation is a way of representing a string value or creating a string.

In my program, a greeting function is built which consists of string value which displays “welcome to the application” at the beginning when the program is opened. Also, a “GreetingAtEnd” function is called which displays string value “thank you” at the end of the program. An empty string actualBinaryNum is created in reverse function which returns a string representation of whatever value was passed in. And, when the user is asked to enter the two decimal numbers, is the input is not between 0 and 255 then the string value “Invalid” is displayed. Also, the input should be in numbers if any string value is provided then other string value is displayed with says “Invalid, enter again”.

The program was developed with six modules. Except main module all of these five modules have its own function which has the major representation to build the program. The main module contains the final setup for the program. The program is developed in a way where it asks the user to enter the first input which is in decimal number system. The input is converted in binary number system from the conversion function. Then the binary converted number is reversed from the reverse function. And, so to the second input as well. The binary_addition function then works for the addition of these two binary numbers and the output is obtained. The main module has the setup for the displaying the final output i.e. the binary addition of the two inputs entered by the user. But, it also have the information to display the respective message when invalid number or invalid data type value is entered as the input. Then, the user is asked if they want to repeat the program or not. The program continues to run or ends according to the choice of the user.

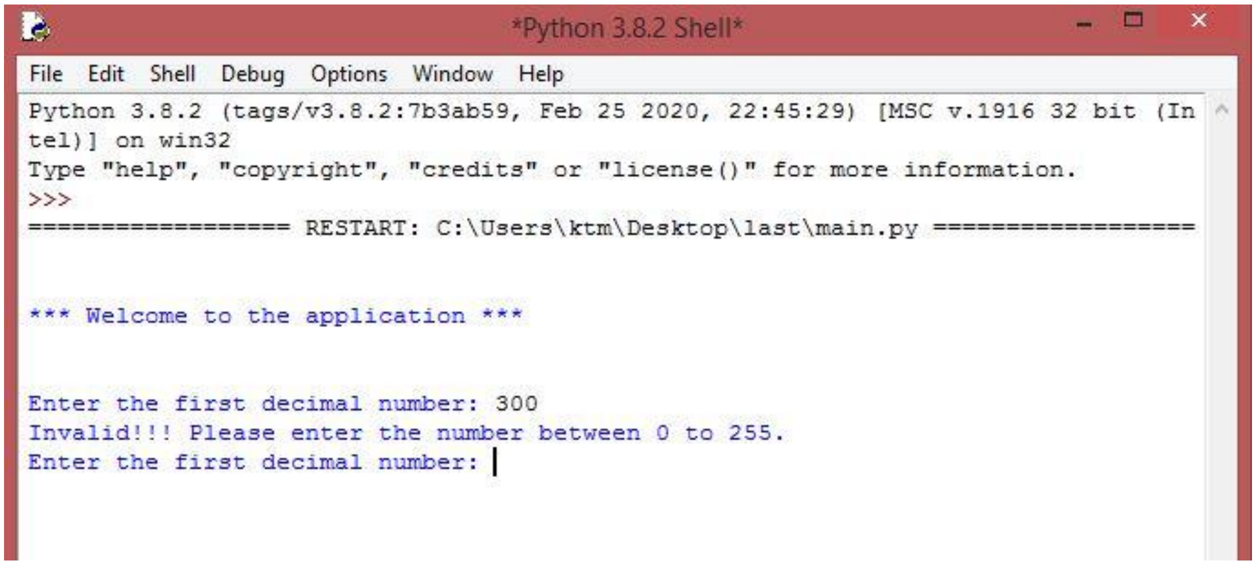
TESTING

Test 1 – Invalid greater value provided as input

Objective	To check the output when decimal number greater than 255 i.e. (invalid value) is entered for the first input value.
Action	Number 300 is entered in the first input value.
Expected Result	An invalid message should be displayed.
Actual Result	An invalid message was displayed.

Conclusion	Test pass.

Table 1: Testing invalid greater value as input



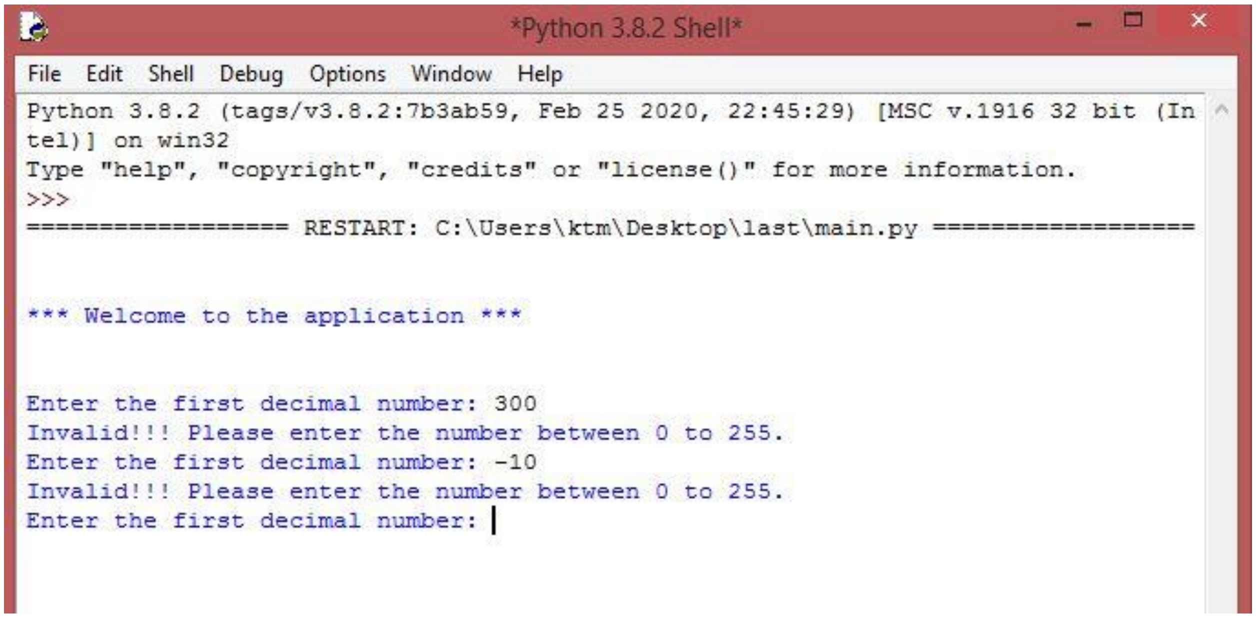
```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: C:\Users\ktm\Desktop\last\main.py ==================


*** Welcome to the application ***


Enter the first decimal number: 300
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: 
```

Figure 5: Invalid greater number as input

Test 2 – Invalid negative value provided as input

Objective	To check the output when decimal number smaller than 0 i.e. (invalid value) is entered for the first input value.
Action	Number -10 is entered in the first input value.
Expected Result	An invalid message should be displayed.
Actual Result	An invalid message was displayed.
Conclusion	Test pass.

Table 2: Testing invalid negative value as input



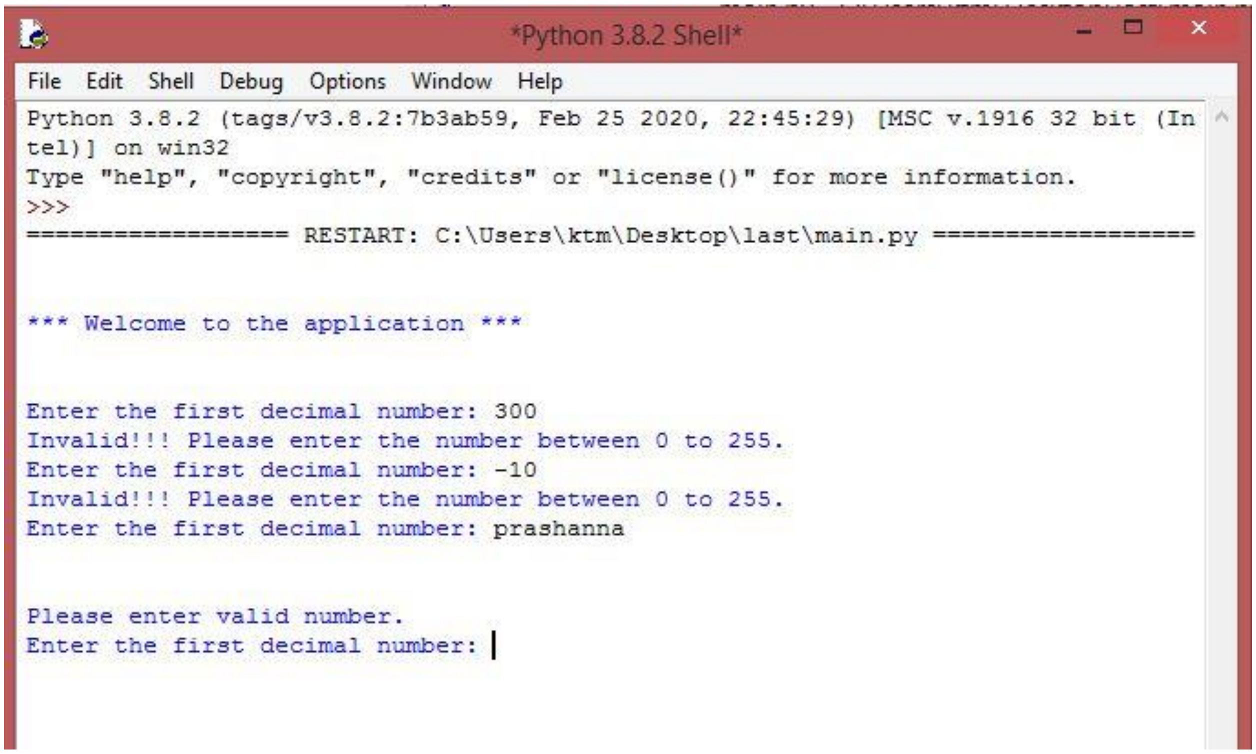
```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
================== RESTART: C:\Users\ktm\Desktop\last\main.py ==================


*** Welcome to the application ***


Enter the first decimal number: 300
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: -10
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: 
```

Figure 6: Invalid negative value provided as input

Test 3 – Invalid string value provided as input

Objective	To check the output when string value i.e. (invalid value) is enter in the first input value.
Action	The string value “prashanna” is entered in the first input value.
Expected Result	An invalid message should be displayed.
Actual Result	An invalid message was displayed.
Conclusion	Test pass.

Table 3: Testing invalid string value as input



```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: C:\Users\ktm\Desktop\last\main.py ==================


*** Welcome to the application ***


Enter the first decimal number: 300
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: -10
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: prashanna


Please enter valid number.
Enter the first decimal number: 
```

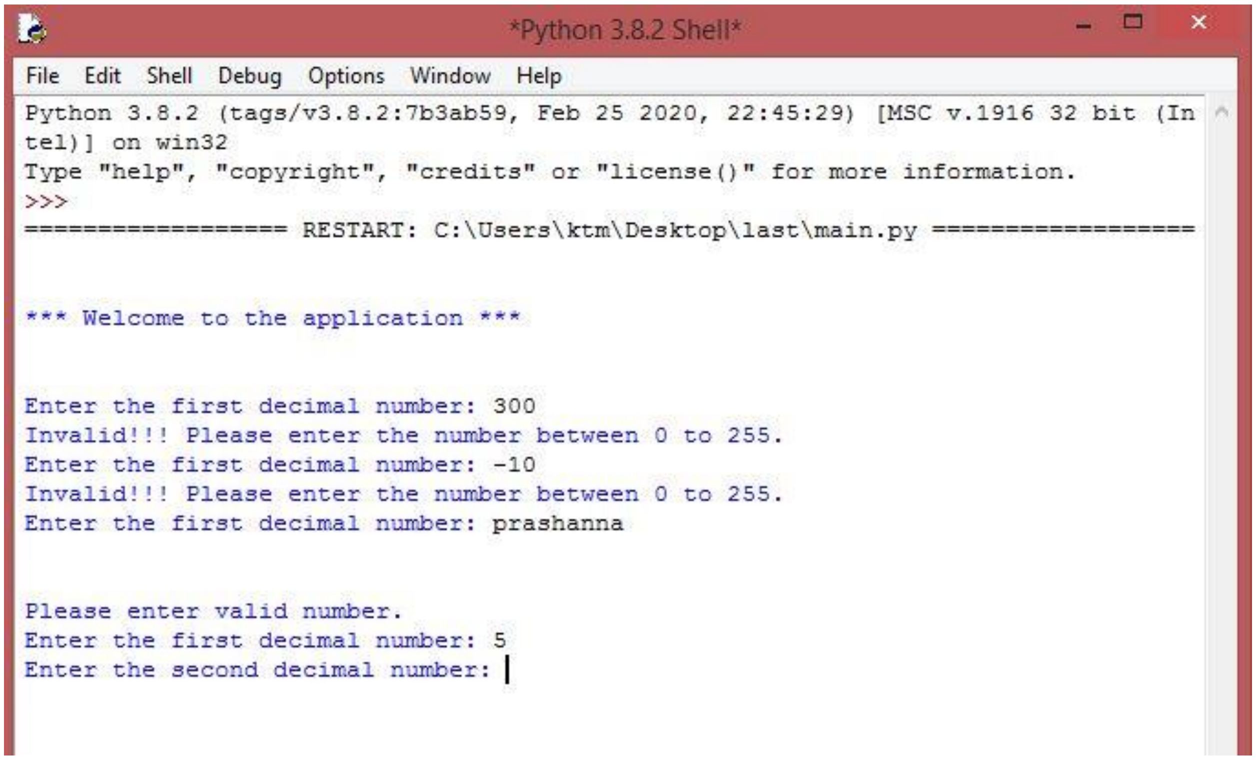
Figure 7: Invalid string value provided as input

Test 4 – Valid value provided as input

Objective	To check the when valid decimal number is entered for the first input value.
Action	Number 5 is entered in the first input value.
Expected Result	The program should continue and ask to enter second decimal number to the user.
Actual Result	The second decimal number was asked to enter to the user.

Conclusion	Test pass.

Table 4: Testing valid value as input



```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
================== RESTART: C:\Users\ktm\Desktop\last\main.py ==================


*** Welcome to the application ***


Enter the first decimal number: 300
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: -10
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: prashanna


Please enter valid number.
Enter the first decimal number: 5
Enter the second decimal number: 
```

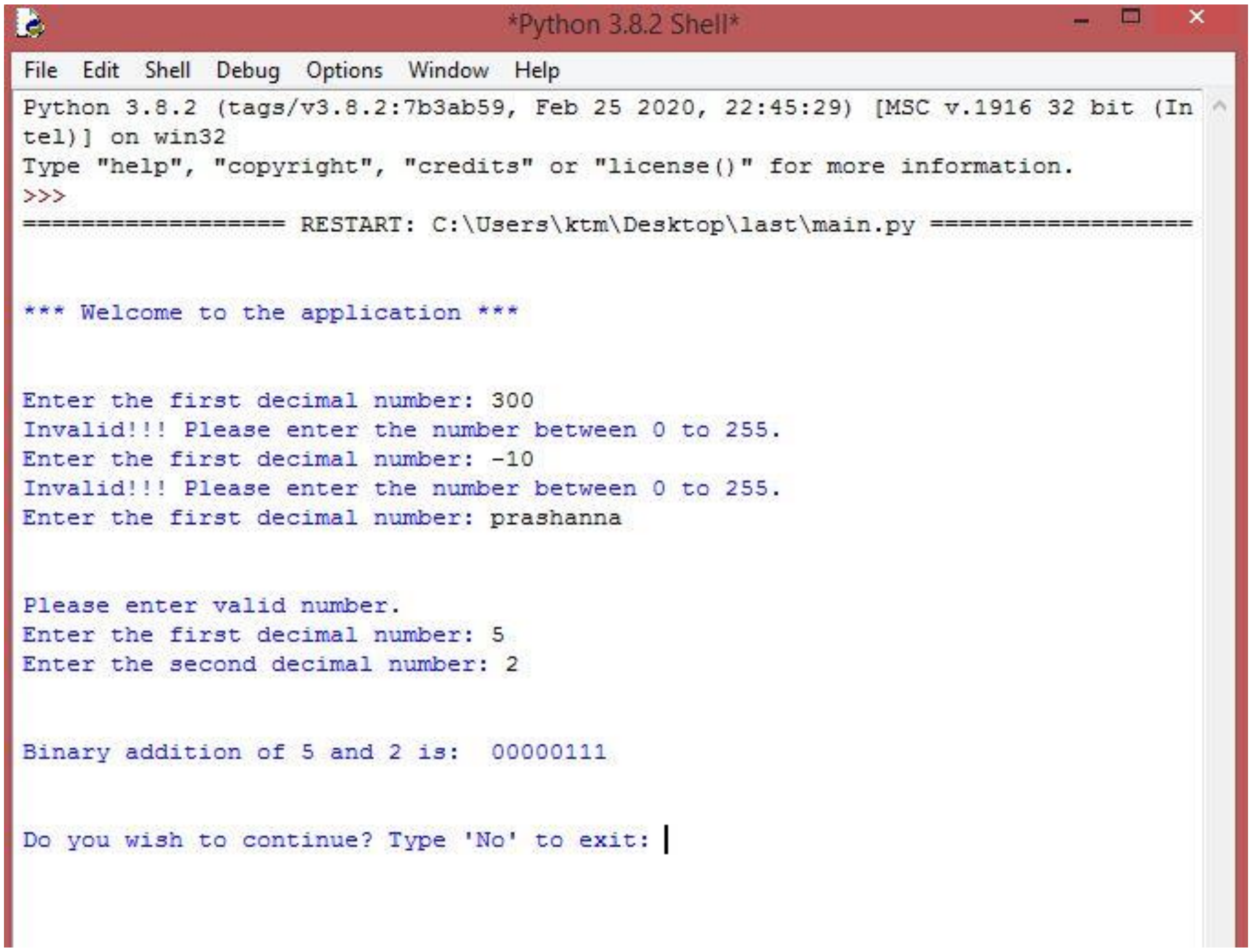
Figure 8: Valid value provided as input

Test 5 – Valid value provided as second input

Objective	To check the when valid decimal number is entered for the second input value.
Action	Number 2 is entered in the second input value.
Expected Result	The program should show the binary addition of the two

	numbers to the user.
Actual Result	The binary addition of the two numbers was displayed to the user successfully.
Conclusion	Test pass.

Table 5: Testing valid value as second input



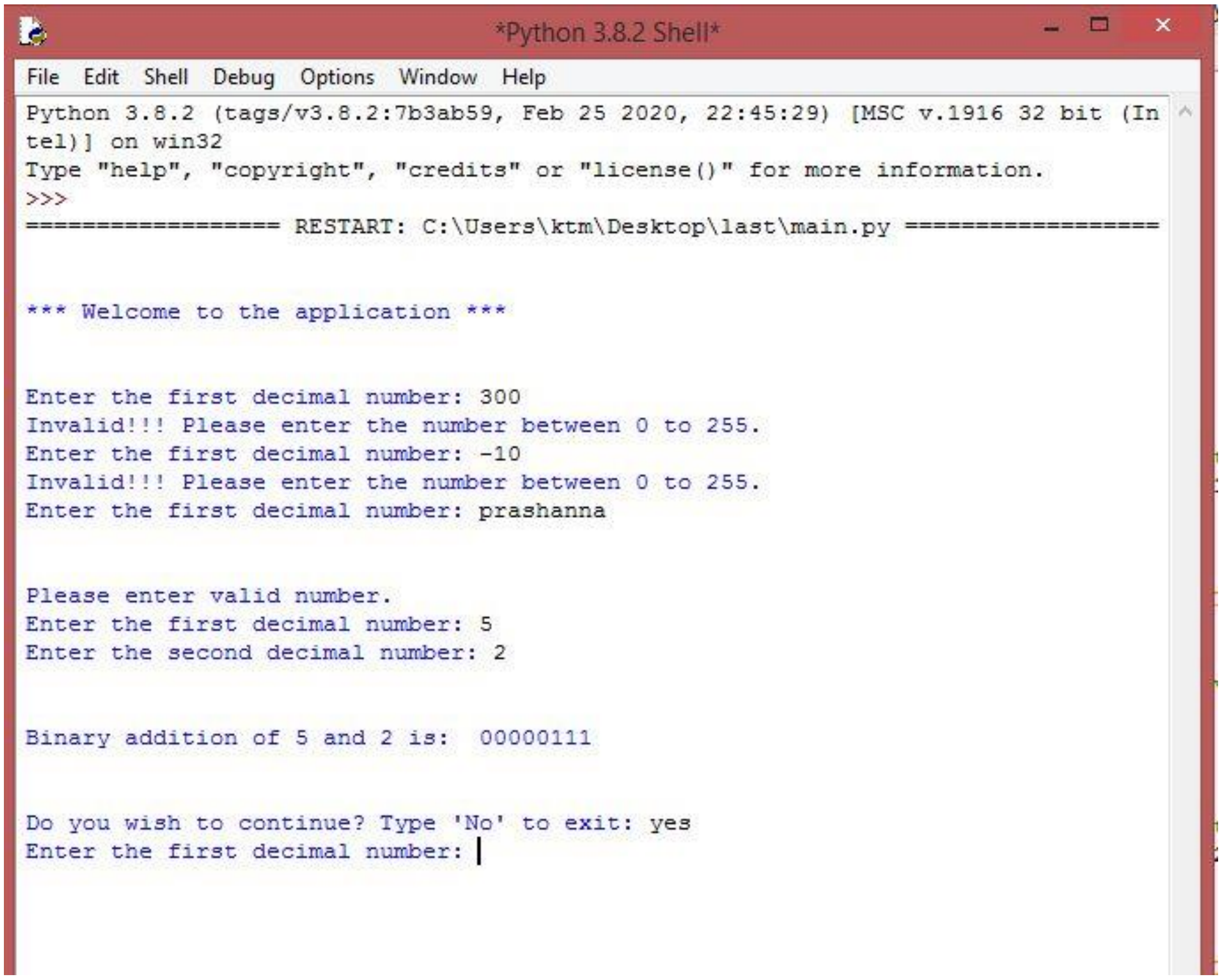
```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: C:\Users\ktm\Desktop\last\main.py ==================


*** Welcome to the application ***


Enter the first decimal number: 300
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: -10
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: prashanna


Please enter valid number.
Enter the first decimal number: 5
Enter the second decimal number: 2


Binary addition of 5 and 2 is:  00000111


Do you wish to continue? Type 'No' to exit: 
```

Figure 9: Valid value provided as second input

Test 6 – Running the program in loop

Objective	To check if the program can continue to run in loop or not according to the users wish.
Action	A “yes” was entered to run the program again.
Expected Result	The program should run again in loop.
Actual Result	The program continued to run again successfully.
Conclusion	Test pass.

Table 6: Testing to run the program in loop



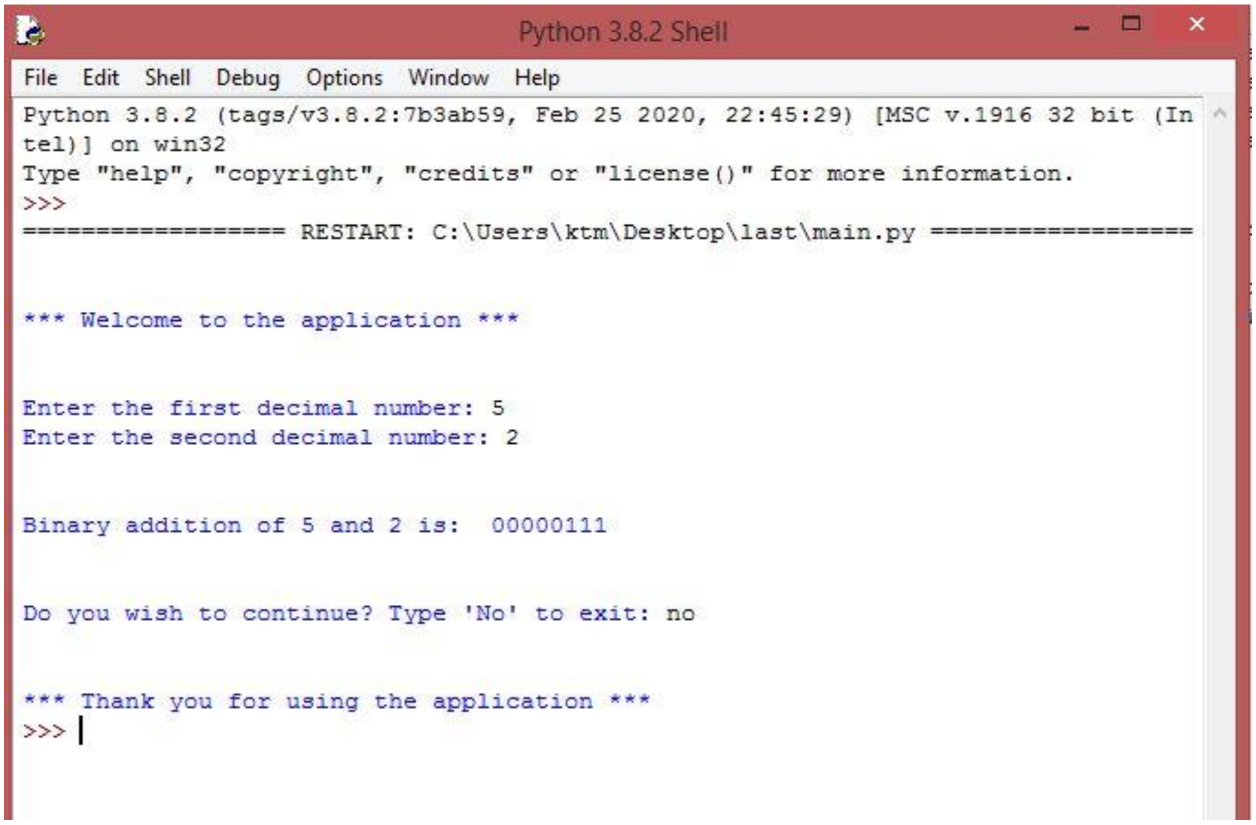
```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: C:\Users\ktm\Desktop\last\main.py ==================


*** Welcome to the application ***


Enter the first decimal number: 300
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: -10
Invalid!!! Please enter the number between 0 to 255.
Enter the first decimal number: prashanna


Please enter valid number.
Enter the first decimal number: 5
Enter the second decimal number: 2


Binary addition of 5 and 2 is:  00000111


Do you wish to continue? Type 'No' to exit: yes
Enter the first decimal number: 
```

Figure 10: Running the program in loop

Test 7– Ending the program

Objective	To check if the program can end or not when the user wants to end it.
Action	A “no” was entered to finish the program.
Expected Result	The program should end displaying a “thank you” message.
Actual Result	The program ended successfully.
Conclusion	Test pass.

Table 7: Testing to end the program



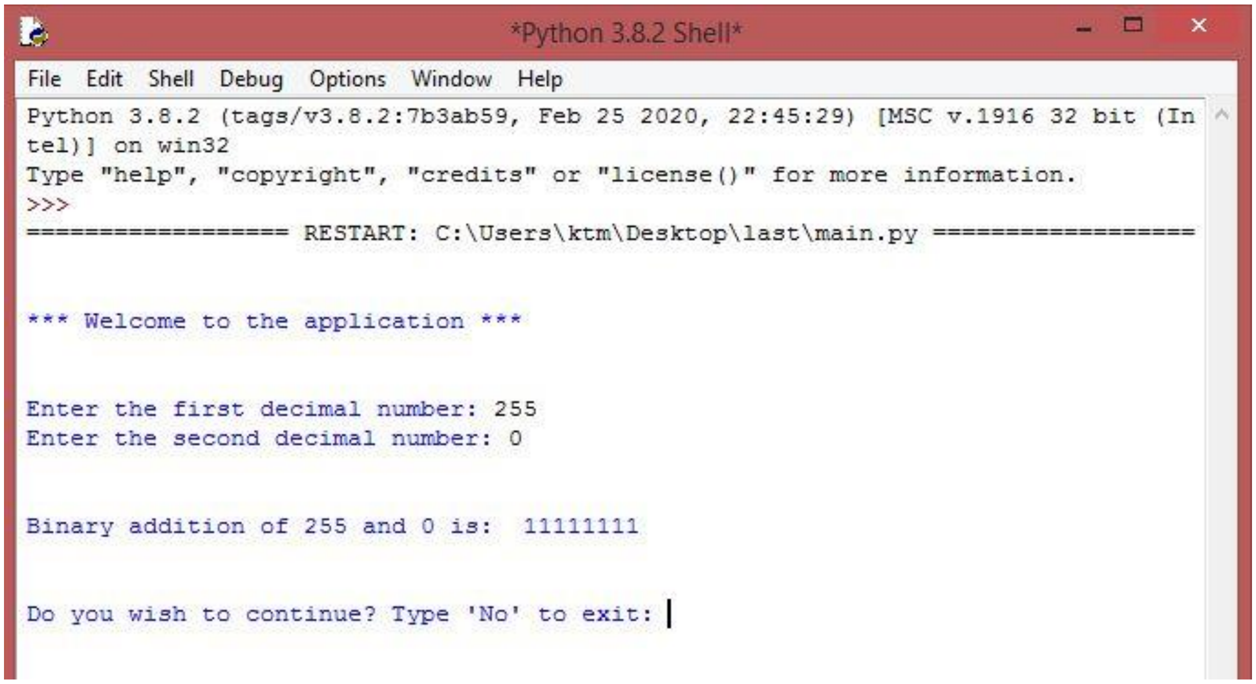
```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: C:\Users\ktm\Desktop\last\main.py ==================


*** Welcome to the application ***


Enter the first decimal number: 5
Enter the second decimal number: 2


Binary addition of 5 and 2 is:  00000111


Do you wish to continue? Type 'No' to exit: no


*** Thank you for using the application ***
>>> |
```

Figure 11: Ending the program

Test 8– Maximum and Minimum value as input

Objective	To check the program when maximum and minimum value are provided as input
Action	The first value was entered 255 and second was entered 0.
Expected Result	The program should show the maximum 8 bit value i.e. 11111111.
Actual Result	The binary addition was shown 11111111.
Conclusion	Test pass.

Table 8: Testing maximum and minimum value as input



```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: C:\Users\ktm\Desktop\last\main.py ==================


*** Welcome to the application ***


Enter the first decimal number: 255
Enter the second decimal number: 0


Binary addition of 255 and 0 is:  11111111


Do you wish to continue? Type 'No' to exit:
```

Figure 12: Maximum and Minimum value as input

CONCLUSION

This coursework was all about developing a software application which simulates the behaviour of a digital circuit performing integer addition. It was a quite difficult assignment for me. It made me research about the project in various aspects. So, I was able to learn about various new components. I was able to gain a lot of information bit adder. Though this coursework was of 8 bit adder but while browsing over the internet I'm also able to know about 1 bit, 2 bit and 4 bit adder. I also learned about logical circuit. I was able to establish a 8 bit logical circuit and gained the information about its working.

I didn't find much problem during creating the format of the program using gates and conversion of the inputs, which I thanks to the video tutorials provided by the tutors. But I did find some problem throughout implementing the bit adder process. I had to research a lot for that which was a lot time consuming. I faced with different kinds of errors and exceptions in this assignment but with the help of the internet, online class reference videos, tutors and friends, I was able to complete this assignment and also able to learn a lot. I finally completed the coursework but wasn't able to submit in the original submission date. I took a lot of time for this assignment since I had a lot of confusions with this work and now submitting the work in the extended submission date.

BIBLIOGRAPHY

Anon. (2020) *DataCamp Inc* [Online]. Available from: https://www.datacamp.com/community/tutorials/data-structures-python?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=332602034361&utm_targetid=ds.

Kuhlman, D. (2012) *wikipedia* [Online]. Available from: https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-7-28.

Wolfram. (2020) *Wolfram* [Online]. Available from: https://www.wolfram.com/system-modeler/examples/more/electrical-engineering/8-bit-adder.