



**Module Code & Module Title**

**CC4051NT Fundamental of Computing**

**Assessment Weightage & Type**

**60% Individual Coursework**

**Year and Semester**

**2019-20 Autumn**

**Student Name: Rup Chandra Khatri**

**London Met ID: 19031837**

**College ID: NP05CP4A190138**

**Assignment Due Date: 8th June, 2020**

**Assignment Submission Date: 8th June, 2020**

*I confirm that I understand my coursework needs to be submitted online via Google Classroom 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

[Introduction: 5](#_Toc42549172)

[1. Model 6](#_Toc42549173)

[a. 8-bit adder 6](#_Toc42549174)

[2. Algorithm 10](#_Toc42549175)

[3. Pseudo code 11](#_Toc42549176)

[forInput 11](#_Toc42549177)

[Conversion 17](#_Toc42549178)

[Gates 18](#_Toc42549179)

[Adder 19](#_Toc42549180)

[Main 19](#_Toc42549181)

[4. Flowchart 22](#_Toc42549182)

[5. Testing: 23](#_Toc42549183)

[i. Test no. 1: 23](#_Toc42549184)

[ii. Test no. 2: 23](#_Toc42549185)

[iii. Test no. 3: 24](#_Toc42549186)

[iv. Test no. 4: 25](#_Toc42549187)

[v. Test no. 5: 25](#_Toc42549188)

[6. Data Structures 26](#_Toc42549189)

[Conclusion: 27](#_Toc42549190)

[Figure 1: OR gate 7](#_Toc42549191)

[Figure 2: AND gate 7](#_Toc42549192)

[Figure 3: X-OR gate 8](#_Toc42549193)

[Figure 4: NOT gate 9](#_Toc42549194)

[Figure 5: Model of 8-bit adder 10](#_Toc42549195)

[Figure 6: Flowchart of the sum of two digits using bit adder 22](#_Toc42549196)

[Figure 7: Program Execution 23](#_Toc42549197)

[Figure 8: Error message when wrong input is given 24](#_Toc42549198)

[Figure 9: Result when the input value is greater than 256. 24](#_Toc42549199)

[Figure 10: Result of testing in Test no. 5 25](#_Toc42549200)

[Figure 11: Successful calculation of 2 digits 26](#_Toc42549201)

[Table 1: OR gate truth table 7](#_Toc42549202)

[Table 2: AND gate truth table 7](#_Toc42549203)

[Table 3: X-OR truth table 8](#_Toc42549204)

[Table 4: NOT truth table 9](#_Toc42549205)

[Table 5: Truth table of full adder 10](#_Toc42549206)

[Table 6: Program execution 23](#_Toc42549207)

[Table 7: Test to show error message because of wrong input 23](#_Toc42549208)

[Table 8: Test when the input value is greater than 256. 24](#_Toc42549209)

[Table 9: To check whether the sum is higher than 256 or not. 25](#_Toc42549210)

[Table 10: Testing about the accurate result 26](#_Toc42549211)

# Introduction:

Python is an interpreted, high-level, object-oriented programming language with dynamic meaning. It has gained popularity because of its clear syntax and readability. Python was created by Guido van Rossum, a former resident of the Netherlands, whose favorite comedy group at that time was Monty Python’s Flying Circus. The source code is freely available and open for modification.

This coursework requires developing a software application that stimulates the behavior of a digital circuit performing integer addition and writing a report to describe the model, algorithm, data structures needed as well as the program developed. There are various kinds of tasks to be performed in this coursework. Firstly we have to construct a model of a byte adder assembled using electronic gates based on model of bit adder. Then we have to specify an algorithm for integer addition based on bitwise operations. We also have to select suitable python data structures to represent the information to be processed by the program. Then a program in Python which implements the model of the adder as designed in the previous task should be created. After that program is tested with selected test data and finally a report should be written to represent the work done.

The objectives of the project are as follows:

1. To know the concept of bit adder.
2. To learn properly about the designing part.
3. To design OOPs with python classes.

The goals of the project are listed below:

1. To improve our programming capability.

The features/characteristics of the python are as follows:

1. It allows more run-time flexibility.
2. Python is a platform independent scripted language with full access to software API's.
3. For building large applications, Python may be compiled to byte-code.
4. It supports interactive mode that enables interacting Testing and debugging of snippets of code.

# Model

## 8-bit adder

8-bit adders are the digital circuits which are usually used in computers for some basic arithmetic operations. We have constructed an 8-bit adder circuit which takes the 2 8-bit integers to calculate the sum for our project.

1. **Logic gates:**

Logic gates are the electronic circuits having one or more than one input and only one output. According to question AND, OR, XOR and NOT gates are described below:

1. **OR gate**

In this gate, if one of the two inputs is higher i.e. 1 then the output will also be high. Otherwise, the output will be 0.

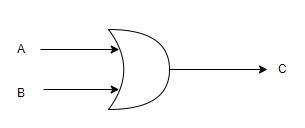


Figure 1: OR gate

**Truth table**

|  |  |  |
| --- | --- | --- |
| A(Input) | B(Input) | C(Output) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Table 1: OR gate truth table

1. **AND gate**

In this gate, if any one of the input is lower i.e. 0 then the output will also be 0. But if the both inputs are high then the output will also be high.

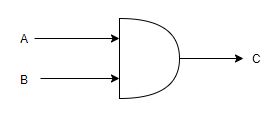


Figure 2: AND gate

**Truth table**

|  |  |  |
| --- | --- | --- |
| A(Input) | B(Input) | C(Output) |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Table 2: AND gate truth table

1. **X-OR gate:**

In this gate, if both inputs are same then the output will be lower i.e. 0. But if the both inputs are different then the output will be higher i.e. 1.

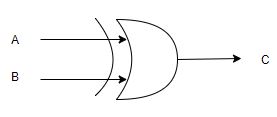


Figure 3: X-OR gate

**Truth table**

|  |  |  |
| --- | --- | --- |
| A(Input) | B(Input) | C(Output) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Table 3: X-OR truth table

1. **NOT gate:**

In this gate, the output is high when the input is low and the output is low when the input is high.

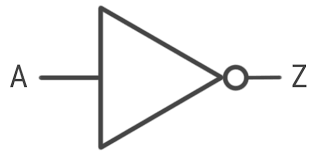


Figure : NOT gate

**Truth table**

|  |  |
| --- | --- |
| Input | Output |
| 1 | 0 |
| 0 | 1 |

Table : NOT truth table

**Process of operation carried out**

Usually we can find 2 types of adder. They are Half-adder and Full- adder. But in this project we have used full adder and its process is described below:

Full-adder is the logical circuit that performs addition operation on 3 binary digits. It also generates a carry out to the next addition column just like the half-adder. Carry-in is the possible carry from the less significant digit. Likewise, carry-out represents a carry to a more notable digit. In the several ways, full-adder can be thought as the 2 half-adders connected with each other with the 1st half-adder passing its carry to other 2nd half-adder.

**Truth table of full adder**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Carry-in | A | B | C (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 |

Table 5: Truth table of full adder

**Diagram**

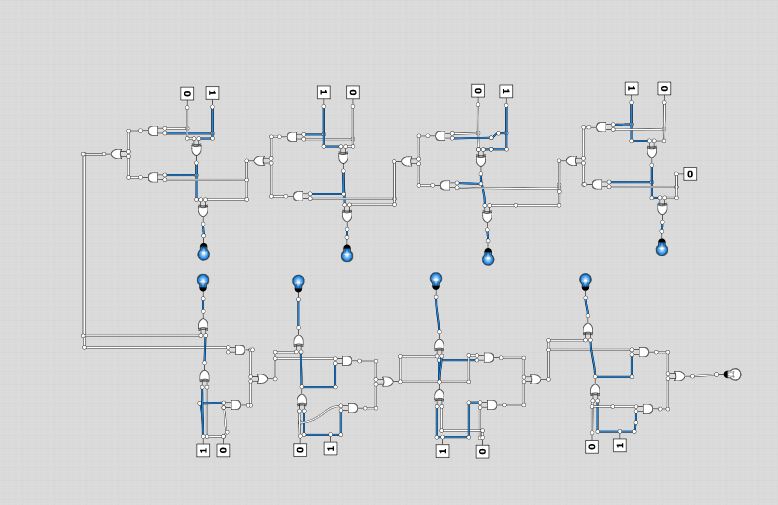


Figure 5: Model of 8-bit adder

# Algorithm

Algorithm is the process of solving the calculation problems in certain number of steps. The algorithm of this project is mentioned below:

Step 1: Start

Step 2: Ask the user to enter ‘d’ for decimal calculation or ‘b’ for binary calculation.

Step 3: If user enters ‘d’:

Step 3.1: Ask the 1st decimal number and store it in num\_1.

Step 3.2: Ask the 2nd decimal number and store it in num\_2.

Step 3.3: Convert num\_1 into binary and store in bin\_1.

Step 3.4: Convert num\_2 into binary and store it in bin\_2.

Step 4: If user enters ‘b’:

Step 4.1: Ask the 1st binary number and store it in str\_1.

Step 4.2: Ask the 2nd binary number and store it in str\_2.

Step 4.3: Convert the str\_1 into binary and store it in bin\_1

Step 4.4: Convert str\_2 into binary and store it in bin\_2

Step 5: Add both bin\_1 and bin\_2 using the gates of byte adder and store it in sum.

Step 6: Display the sum

Step 7: Stop

# Pseudo code

## forInput

INCLUDE conversion

FUNCTION DecimalInput()

PRINT “Enter 1st decimal number.”

PRINT “Note: The number should lie between 0 and 256.”

WHILE true

WHILE true

TRY

num\_1= INT (INPUT( “Enter the 1st decimal number.”))

IF num\_1is less than 0 or num\_1 is greater than 255

PRINT “Error !”

PRINT “The number should lie between 0 and 255.”

CONTINUE

ELSE

BREAK

BREAK

EXPECT ValueError

PRINT “ Error! ”

PRINT “Please enter the valid input”

CONTINUE

WHILE true

TRY

num\_2= INT (INPUT “Enter the 2nd decimal number:”)

IF num\_2 is smaller than 0 or num\_2 is greater than 255

PRINT “Error”

PRINT “The number should lie between 0 and 256.”

CONTINUE

ELSE:

BREAK

BREAK

EXPECT ValueError

PRINT “Error!”

PRINT “Please enter valid input”

CONTINUE

IF the sum of num\_1and num\_2 is greater than 255

PRINT “Error!”

PRINT “The sum of two numbers should be less than 255”

CONTINUE

ELSE:

BREAK

RETURN num\_1, num\_2

FUNCTION binInput()

PRINT “Enter the binary numbers”

PRINT “Note: The binary number should be maximum at 1111111 and minimum at 00000000.”

WHILE true

WHILE true

num\_1= INPUT “Enter the 1st binary number: “

size = len(num\_1)

IF num\_1is empty

PRINT ”Error!”

CONTINUE

ELIF size is greater than 8

PRINT “Error!!!”

PRINT “The entered binary number should be of 8 bits”

CONTINUE

ELSE

TRY

num\_1 = INT(num\_1)

li = LIST(MAP(INT,STR(num\_1)))

sets = SET(li)

IF sets is equal to {0} or {1} or {0,1}

BREAK

ELSE

PRINT “Error!”

PRINT “The input value must be 0-1”

CONTINUE

EXCEPT ValueError

PRINT “Error! Invalid input.”

PRINT “Alphabet, character and negative value is not allowed. “

CONTINUE

WHILE true

num\_2= INPUT “Enter the 2nd binary number: “

length = len(num\_2)

IF secondNumber is empty

PRINT “Error!”

PRINT “fill the field.”

CONTINUE

ELIF size is greater than 8

PRINT “Error!”

PRINT “The entered binary number should be of 8 bits”

CONTINUE

ELSE

TRY

Num\_2 = INT(num\_2)

li = LIST(MAP(INT,STR(num\_2)))

sets = SET(li)

IF sets is equal to {0} or {1} or {0,1}

BREAK

ELSE

PRINT “Error!”

PRINT “The input value should be 0-1”

CONTINUE

EXCEPT ValueError

PRINT “Error! Given input is invalid.”

PRINT “Alphabet, chatacter and negative value is not allowed “

CONTINUE

x = stringToList(num\_1)

y = stringToList(num\_2)

m = binaryToDecimal (x)

n = binaryToDecimal (y)

IF sum of m and n is greater than 255

PRINT “Error!”

PRINT “The sum of two binary number is larger than byte. Please, try again.”

BREAK

ELSE

BREAK

RETURN num\_1, num\_2

## 

## Conversion

FUNCTION decToBin(decimal)

list\_1 = [0, 0, 0, 0, 0, 0, 0, 0]

i = 7

WHILE decimal is greater than 0

remainder = decimal modulo 2

list\_1 [i] = remainder

decimal = INT(decimal / 2)

i = i – 1

return list\_1

FUNCTION listToStr list\_1)

binary = “ ”

FOR b IN range (8)

binary = binary + str list\_1 [b])

return binary

FUNCTION strToList(num)

bin\_1 = [0, 0, 0, 0, 0, 0, 0, 0]

a = 7

FOR i IN range(LEN(num)-1, -1, -1)

bin\_1[n] = INT(num[i])

a = a – 1

return bin\_1

FUNCTION binaryToDecimal(binary)

count= 0

i = 7

decimal = 0

WHILE i is greater or equals to 0

decimal = decimal + (binary[i] \* (2 \*\* count))

count = count+ 1

i = i – 1

return decimal

## Gates

FUNCTION orGate(num\_1, num\_2)

RETURN num\_1 | num\_2

FUNCTION andGate(num\_1, num\_2)

RETURN num\_1 & num\_2

FUNCTION xorGate(num\_1, num\_2)

RETURN num\_1 ^ num\_2

## Adder

INCLUDE gates

FUNCTION binaryAdder(f\_1, f\_2)

binary = [0, 0, 0, 0, 0, 0, 0, 0]

i = 7

carry = 0

WHILE i greater or equals to 0

num\_1 = f\_1 [i]

num\_2 = f\_12[i]

sum\_1= xorGate (xorGate (num\_1, num\_2), carry)

binary [i] = sum\_1

bit\_1 = andGate (xorGate (num\_1, num\_2), carry)

bit\_2 = andGate (num\_1, num\_2)

carry = orGate (bit\_1, bit\_2)

i = i – 1

RETURN binary

## Main

INCLUDE forInput

INCLUDE conversion

INCLUDE adder

WHILE true

x = INPUT “Enter: ‘D’ for Decimal and ‘B’ for Binary. Enter the suitable letter.”

IF x = “D”

num\_1, num\_2 = DecimalInput()

bin\_1 = decimalToBin(num\_1)

bin\_2 = decimalToBin(num\_2)

result = binaryAdder(bin\_1, bin\_2)

ELIF x = “B”

str\_1, str\_2 = BinaryInput()

bin\_1 = strToList(str\_1)

bin\_2 = strToList(str\_2)

result = binAdder(bin\_1, bin\_2)

num\_1 = binToDec(bin\_1)

num\_2 = binToDec(bin2)

ELIF x is empty

PRINT “Error!”

PRINT “The field is empty. You are requested to fill the field properly..”

CONTINUE

ELSE

PRINT “Error! Invalid input.”

PRINT “Please fill the field with given suitable input.”

CONTINUE

PRINT “1st binary value : “, listToStr(bin\_1)

PRINT “2nd Binary value: “, listoStr(bin\_2)

PRINT “The sum of two binary value is: “listToStr(result)

PRINT “1st decimal value: “, num\_1

PRINT “2nd decimal value: “, num\_2

PRINT “The sum of two decimal value is: “ num1 + num2

PRINT “Do you want to again continue the program?”

WHILE true

PRINT “Enter 'C' to continue and 'E' to exit the program..”

y = INPUT “Enter: “

IF y = “C” or “E”

BREAK

ELIF y is empty

PRINT “Error!”

PRINT “Please fill the field with valid input.”

CONTINUE

ELSE

PRNT “Error!”

PRINT “Please enter the valid input..”

CONTNUE

IF y= ‘C’

CONTINUE

ELSE

BREAK

# Flowchart

It is the diagrammatic representation of an algorithm, a step-to-step approach to solve a task.

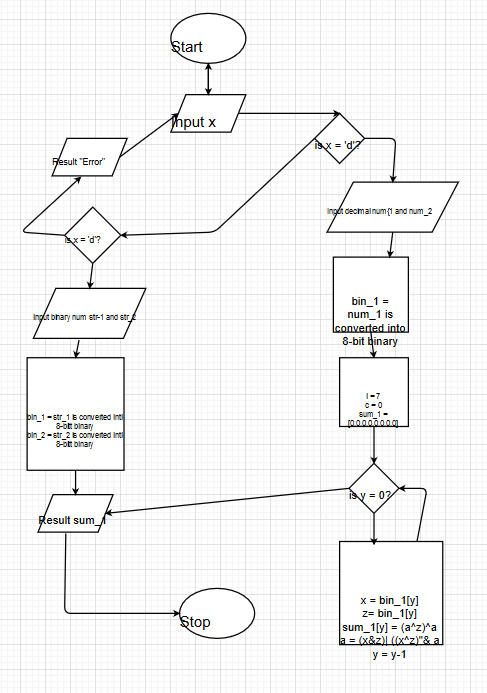


Figure 6: Flowchart of the sum of two digits using bit adder

# Testing:

## Test no. 1:

|  |  |
| --- | --- |
| Action | The main.py file is run. |
| Expected result | To show whether the command ‘Enter ‘d’ for decimal or ‘b’ for binary’ appears or not when executed. |
| Actual result | Enter ‘d’ for decimal or ‘b’ for binary appeared when main.py executed. |
| Conclusion | Well, test got failed. |

Table : Program execution



Figure : Program Execution

## Test no. 2:

|  |  |
| --- | --- |
| Action | Wrong input is given when the program asks the user to enter ‘D’ for decimal or ‘B’ for binary. |
| Expected Result | Error message should be displayed saying “Error! Please, enter the suitable value”. |
| Actual result | The same error message got displayed as described in the expected result when wrong input is given. |
| Conclusion | The test got failed. |

Table : Test to show error message because of wrong input

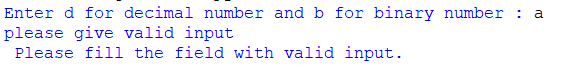


Figure : Error message when wrong input is given

## Test no. 3:

|  |  |
| --- | --- |
| Action | Value greater than 256 is assigned. |
| Expected Result | Error message should come saying ‘Error! Please enter the number that is less than 256.’ |
| Actual result | Error message came when the higher value was assigned. |
| Conclusion | Test failed. |

Table 8: Test when the input value is greater than 256.

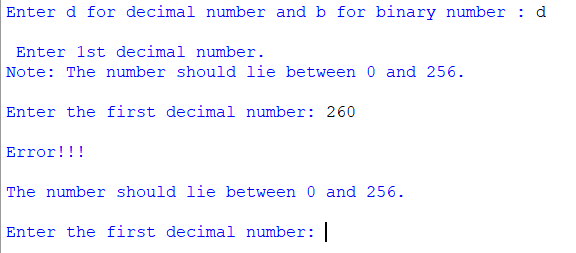


Figure : Result when the input value is greater than 256.

## Test no. 4:

|  |  |
| --- | --- |
| Action | Two high digit numbers are assigned to perform the calculations. |
| Expected result | Error message should be displayed. |
| Actual result | Error message is displayed saying ‘Error! The sum of 2 numbers should be less than 256.’ |
| Conclusion | Test failed. |

Table : To check whether the sum is higher than 256 or not.

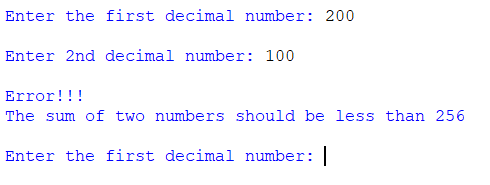


Figure : Result of testing in Test no. 5

## Test no. 5:

|  |  |
| --- | --- |
| Action | 2 decimal numbers where given to perform the calculation. |
| Expected Result | Accurate result should be printed as a result. |
| Actual result | Accurate result was printed with no any error. |
| Conclusion | Test failed. |

Table : Testing about the accurate result

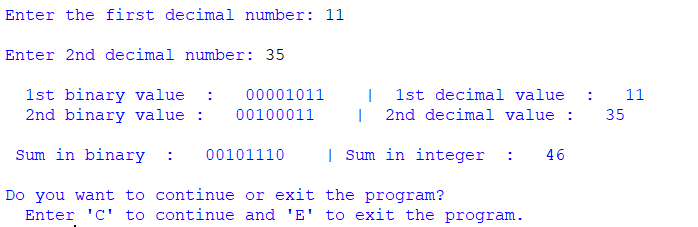


Figure : Successful calculation of 2 digits

# Data Structures

Data structure is the particular way of organizing or managing the data in a computer so that it can be used successfully.

The types of data structures used in the program are as follows:

1. **Strings:** They are the arrays of bytes that represents Unicode characters. It is the collection of one or more characters put in single or double quotes.
2. **Int**: It is represented by the int class. It contains either positive or negative whole numbers. There is no limit to how long integer value can be.
3. **Boolean:** It is the data type that has one of the two built-in values i.e. True or False. T and F are the valid Booleans otherwise python gives away the error.
4. **Sets**: Sets are the unordered or unmanaged collection of the unique objects. It is immutable i.e. no any other data can’t be added to them when they are once created. Sets are represented by using () symbol.
5. **List:** It can be used for any type of object. It is accessed like the strings. Hence, they are easy to use and they grow and shrink automatically as they are used.

# Conclusion:

This project is about programming language i.e. python. As we are newly to this programming it is a little bit hard for writing at first. In this work we have to reflect model based on bit adder, algorithm, data structures, program, its testing and report writing.

In this coursework, we have to develop small kind of software. All the work such as model, algorithm, data structures, program and testing are done and presented. Programs are done in five different IDLE platform. Each platform has its own importance. All other are part of documentation. All the works were done according to the question.