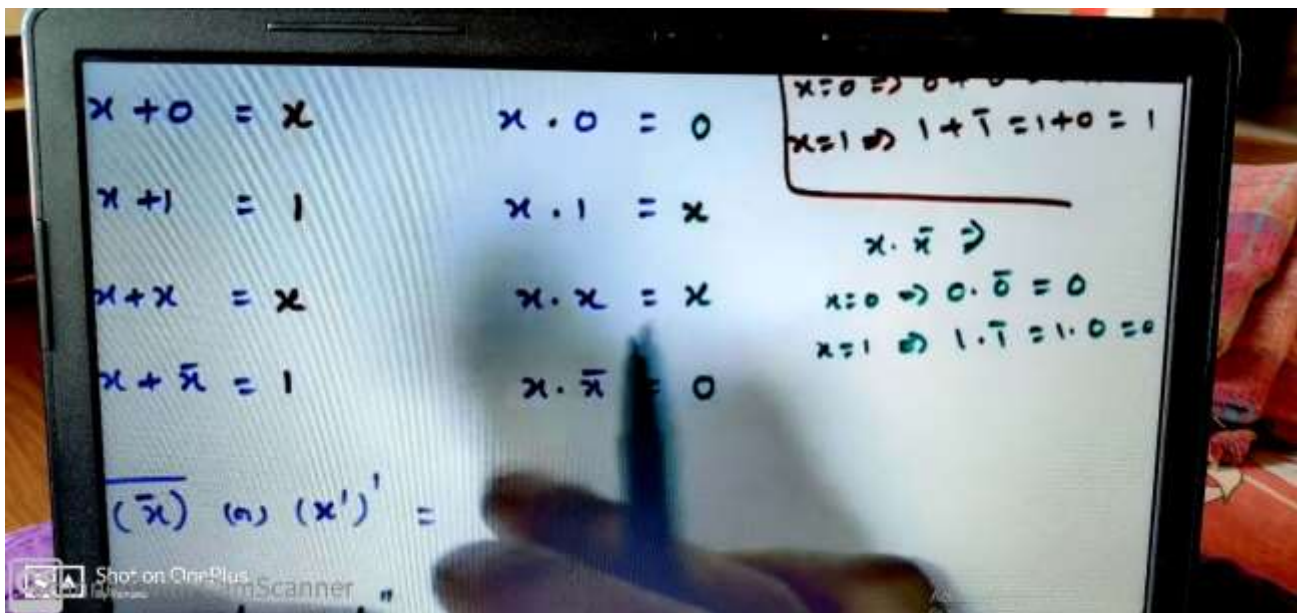
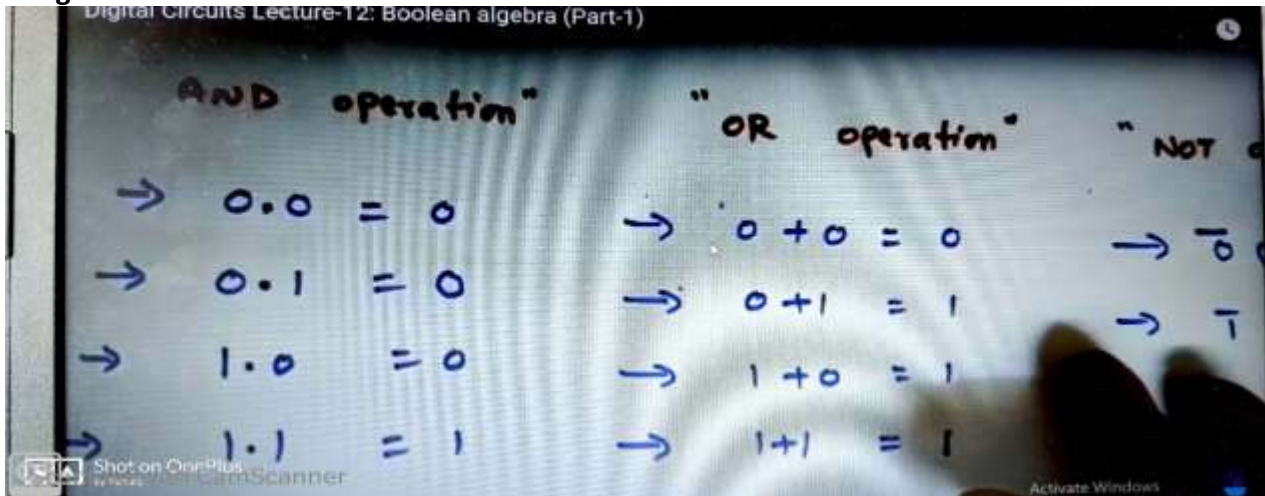


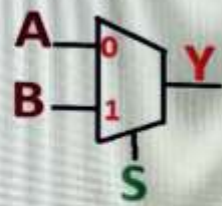
## DAILY ASSESSMENT FORMAT

Date:	28-05-2020	Name:	Roshni A B
Course:	Logic Design	USN:	4AL17EC080
Topic:	Boolean equations for digital circuit. Combinational circuits: conversion of MUX and decoders to logic gates. Design 7 segment decoder with common anode display.	Semester & Section:	6 <sup>th</sup> B sec
Github Repository:	roshni-online		

### FORENOON SESSION DETAILS

Image of session





SELECTION (S)	OUTPUT(Y)
0	A
1	B

$$Y = A\bar{S} + BS$$

$$2^n - 1$$

$$2^n = \text{inputs}$$

$n$  = selection lines

BCD to 7 segment decoder

BCD to 7-segment decoder



A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

## Logic Design

### Boolean equations for digital circuits

- Cost of the circuit
- Simple realization of a circuit
- Boolean algebra is a system of mathematical
- It is defined with set of elements, a set of operators, and a number of axioms or postulates
- set of elements (0, 1)
- binary operators - or, and
- unary " - NOT

### Axioms & Laws of Boolean Algebra

Difference between boolean algebra, ordinary algebra, and binary number system

#### Boolean Algebra :-

$$A + A = A \quad A \cdot A = A$$

$$1 + 1 = 1 \quad 1 \cdot 1 = 1$$

#### Ordinary Algebra

$$A + A = 2A \quad A \cdot A = A^2$$

$$1 + 1 = 2 \quad 1 \cdot 1 = 1$$

#### Binary number system

$$1 + 1 = (10) \quad 1 \cdot 1 = 1$$

### Axioms and Postulates :-

$$\rightarrow (x')' = x$$

#### Identity Element :-

OR operation

additive identity: 0

multiplicative: 1

AND operation

#### Laws of Boolean Algebra :-

$$1) \text{ Commutative Law :- } x + y = y + x \quad x \cdot y = y \cdot x$$

$$A + B = B + A \quad A \cdot B = B \cdot A$$

### MUX to logic gates :-

1) AND, NOR → "Universal gates"  
MUX and Decoders are called "Universal logic"

#### Switcher Design

$$Y = 1 \cdot \bar{A} + 0 \cdot A$$

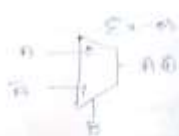
$$Y = \bar{A}$$

#### AND gate

$$Y = 0 \cdot \bar{B} + A \cdot B$$

$$Y = AB$$

#### OR gate



#### Ex - NOR



#### Bcd to 7 segment decoder

	0	1	2	3	4	5	6	7	8	9
a	1	1	1	1	1	1	1	1	1	1
b	1	1	1	1	1	1	1	1	1	1
c	1	1	1	1	1	1	1	1	1	1
d	1	1	1	1	1	1	1	1	1	1
e	1	1	1	1	1	1	1	1	1	1
f	1	1	1	1	1	1	1	1	1	1
g	1	1	1	1	1	1	1	1	1	1

$$a = \bar{c} + A + B \cdot \bar{D} + BD$$

$$b = \bar{b} + CD + \bar{E} \cdot \bar{D}$$

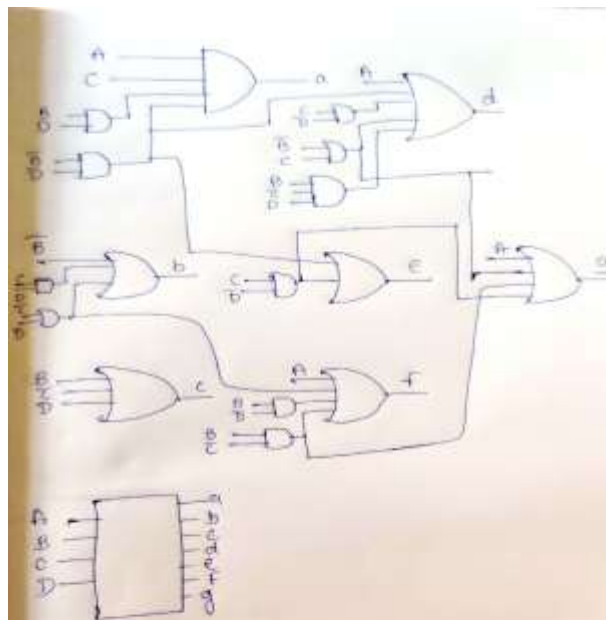
$$c = B + \bar{E} + D$$

$$d = A + B \cdot \bar{D} + C \cdot \bar{D} + BC + B \cdot \bar{E} \cdot D$$

$$e = \bar{B} \cdot \bar{D} + C \cdot \bar{D}$$

$$f = A + \bar{E} \cdot \bar{D} + B \cdot \bar{D} + BC$$

$$g = A + B \cdot \bar{D} + BC + C \cdot \bar{D}$$

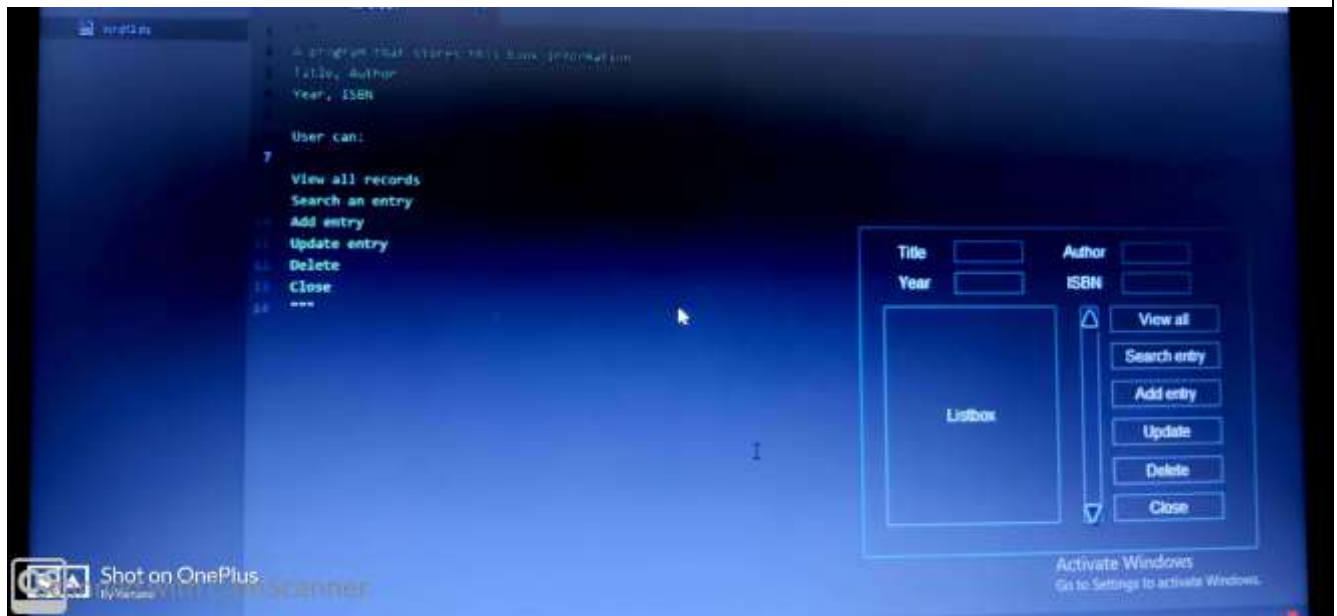
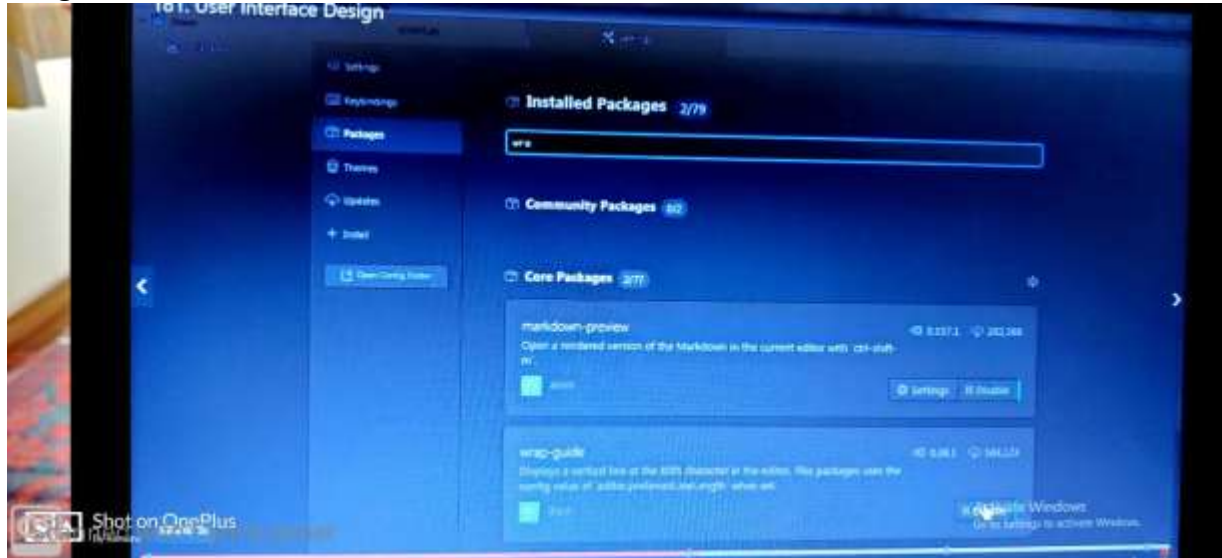




Date:	28-05-2020	Name:	Roshni A B
Course:	Python Programming	USN:	4AL17EC080
Topic:	Application 5: Build a desktop database application.	Semester & Section:	6 <sup>th</sup> sem B sec

#### AFTERNOON SESSION DETAILS

##### Image of session



## 184. Connecting the Frontend to the Backend, Part 1

```

1 from tkinter import *
2 from backend import *
3
4 def view_command():
5     list.delete(0, list.size())
6     for row in backend.view():
7         list.insert(END, row)
8
9 window=Tk()
10
11 list=Listbox(window, text="Title")
12 list.grid(row=0, column=0)
13
14 list=Listbox(window, text="Author")
15 list.grid(row=0, column=1)
16
17 list=Listbox(window, text="Year")
18 list.grid(row=0, column=2)
19
20
21

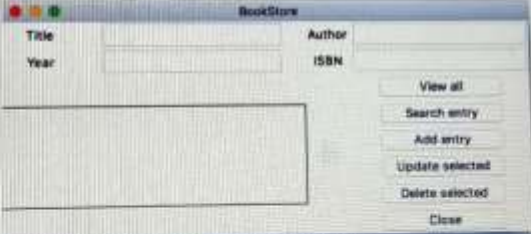
```

Traceback (most recent call last):  
 File "C:\Users\Warius.Grete-PC\AppData\Local\Programs\Python\Python35-32\lib\tkinter\\_init\_.py", line 1549, in \_call\_...  
 return self.func(\*args)  
 File "frontend.py", line 5, in view\_command  
 for row in backend.view():  
 NameError: name 'backend' is not defined  
 PS D:\app\td\_kinter\_sqlite3\Demo> python frontend.py  
 [(4, 'The sea', 'John Tablet', 1918, 913123132), (2, 'The sea', 'John Tablet', 1918, 913123132), (4, 'The Active Life', 'John Tablet', 1918, 913123132)]

## Fixing the Bug (Practice)

### Exercise

If you haven't already noticed, the program has a bug. When the listbox is empty and the user clicks the listbox, an `IndexError` is generated in the terminal:



```

Addition-MD@pcap06-tkinter-sqlite-wsl:~/python3 frontend.py
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\Warius.Grete-PC\AppData\Local\Programs\Python\Python35-32\lib\tkinter\_init_.py", line 1549, in _call_...
  return self.func(*args)
  File "frontend.py", line 4, in get_selected_row
  index=list1.curselection()[0]
  IndexError: tuple index out of range

```

What does this error happen?

## Application 5: Build a Desktop Database

### Application:

A program that stores this book information:

Title: Author

Year: ISBN

User can:

view all records

search an entry

add entry

update entry

Delete

Close

```
import sqlite3
```

```
def connect():
```

```
    conn = sqlite3.connect("books.db")
```

```
    cur = conn.cursor()
```

```
    cur.execute("Create Table if Not exists book
```

```
        (id Integer Primary Key, title text, author
```

```
        text, year integer, isbn integer)")
```

```
    conn.commit()
```

```
    conn.close()
```

```
def insert(title, author, year, isbn):
```

```
    conn = sqlite3.connect("books.db")
```

```
    cur = conn.cursor()
```

```
    cur.execute("Insert into book values (Null, ?, ?, ?, ?)"
```

```
        (title, author, year, isbn))
```

```
    conn.commit()
```

```
    conn.close()
```

```
def view():
```

```
    conn = sqlite3.connect("books.db")
```

```
    cur = conn.cursor()
```

```
    cur.execute("select * from book")
```

```
    rows = cur.fetchall()
```

```
    conn.close()
```

```
    return rows
```

Solution:-

```
def get_selected_row(event):
```

```
    try:
```

```
        global selected_tuple
```

```
        index = list1.curselection()[0]
```

```
        selected_tuple = list1.get(index)
```

```
        e1.delete(0, END)
```

```
        e1.insert(END, selected_tuple[1])
```

```
        e2.delete(0, END)
```

```
        e2.insert(END, selected_tuple[2])
```

```
        e3.delete(0, END)
```

```
        e3.insert(END, selected_tuple[3])
```

```
        e4.delete(0, END)
```

```
        e4.insert(END, selected_tuple[4])
```

```
    except IndexError:
```

```
        pass
```