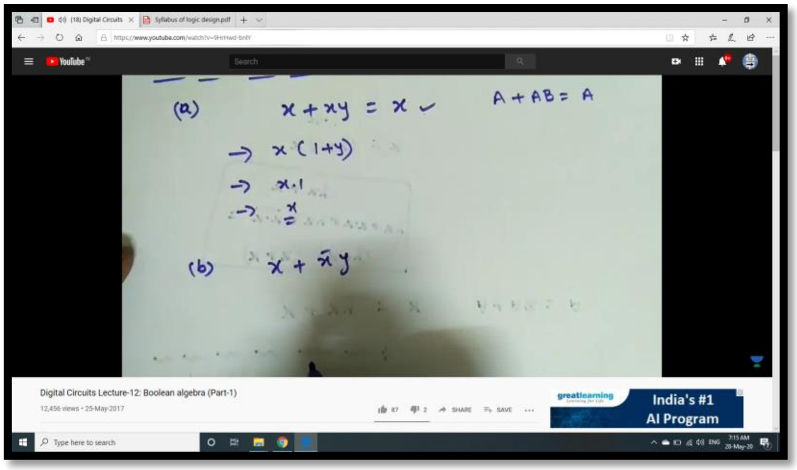
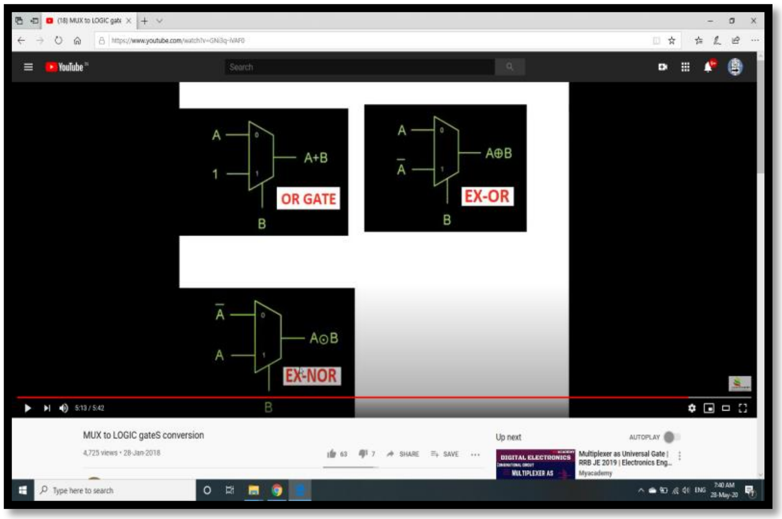
DAILY ASSESSMENT FORMAT

Date:	28/05/2020	Name:	Davis S. Patel
Course:	Logic Design	USN:	4AL16EC045
Topic:	Boolean equations for digital circuits. Combinational circuits: Conversion of MUX and Decoders to logic gates.	Semester & Section:	8th - A
GitHub Repository:	Davis		

FORENOON SESSION DETAILS

Image of session


(a) x + xy = x ✓
A + AB = A
→ x (1+y)
→ x.1
→ x
(b) x + x̄y
Digital Circuits Lecture-12: Boolean algebra (Part-1)
12,456 views • 25-May-2017
India's #1 AI Program


A
1
A+B
OR GATE
B
A
Ā
A⊕B
EX-OR
B
Ā
A
A⊙B
EX-NOR
B
MUX to LOGIC gateS conversion
4,725 views • 28-Jan-2018
Up next
Multiplexer as Universal Gate | RRB JE 2019 | Electronics Eng...

Report –

Boolean algebra:

- In 1854, George Boole Developed an Algebraic System Called Boolean algebra.
- Boolean algebra is a System of Mathematical Logics.
- It is Defined With a set of Elements, a set of Operators and a Number of Postulates

Laws of Boolean algebra:

- **Commutative Law**

  Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.

  $X+Y=Y+X$

  $A+B =B+A$

  $X.Y=Y.X$

  $A.B=B.A$

- **Associative Law**

  If a logical operation of any two Boolean variables is performed first and then the same operation is performed with the remaining variable gives the same result, then that logical operation is said to be Associative.

  $X+(Y+Z)=(X+Y)+3$

  $A+(B+C)=(A+B)+C$

  $X.(Y.Z)=(X.Y).Z$

  $A.(B.C)=(A.B).C$

- **Distributive Law**

  If any logical operation can be distributed to all the terms present in the Boolean function, then that logical operation is said to be Distributive.

  X(Y+Z)=XY+YZ

  A(B+C)=AB+AC

- **Absorption Theorem**

  This law enables a reduction in a complicated expression to a simpler one by absorbing like terms.

  X+XY=X

  A+AB=A

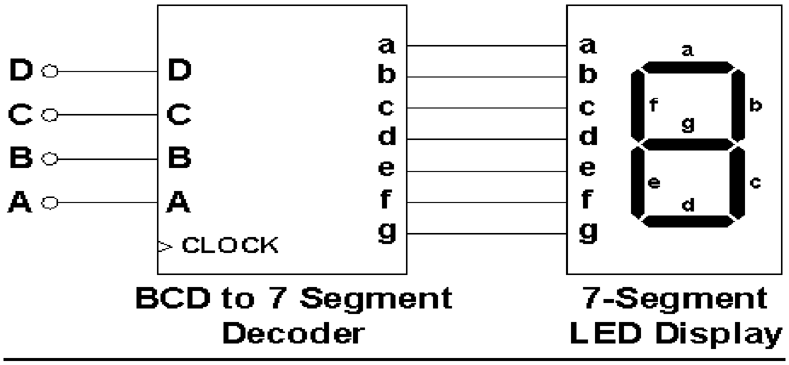
  X+~XY=X+Y

MUX to Logic Gates Conversion:

1. NAND, NOR-Universal Gates
2. Universal Gates
3. MUX and Decoders are Called Universal Logic

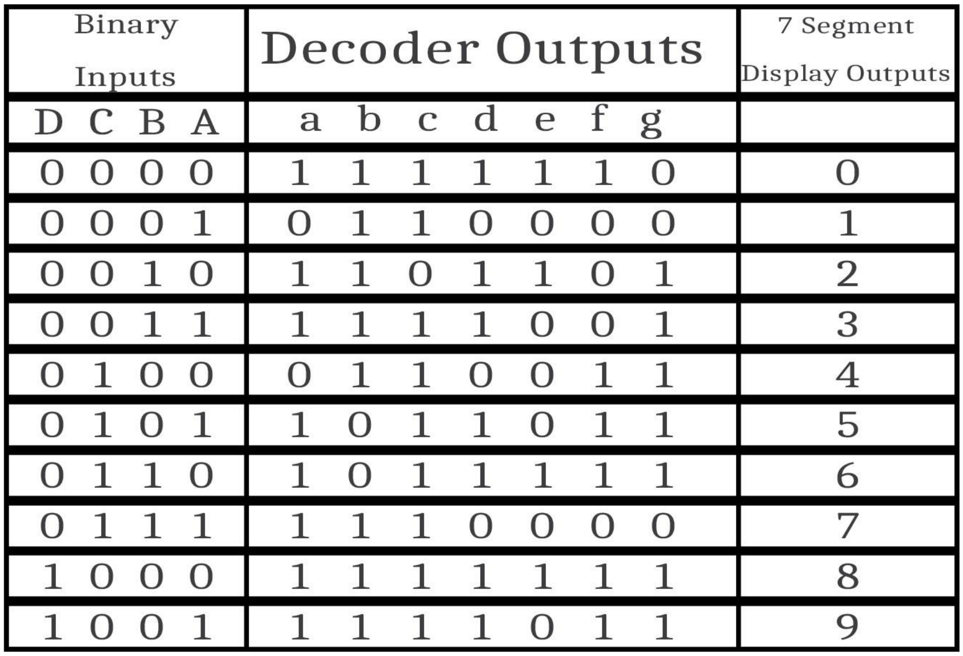
Multiplexer is device which Selects one or Several Digital or Analog Inputs and It will forward it to the Output Line, Which is Single Output line.

BCD to SEVEN Segment Decoder



BCD to 7 Segment Decoder

7-Segment LED Display



Binary Inputs	Decoder Outputs	7 Segment Display Outputs
D C B A	a b c d e f g	
0 0 0 0	1 1 1 1 1 1 0	0
0 0 0 1	0 1 1 0 0 0 0	1
0 0 1 0	1 1 0 1 1 0 1	2
0 0 1 1	1 1 1 1 0 0 1	3
0 1 0 0	0 1 1 0 0 1 1	4
0 1 0 1	1 0 1 1 0 1 1	5
0 1 1 0	1 0 1 1 1 1 1	6
0 1 1 1	1 1 1 0 0 0 0	7
1 0 0 0	1 1 1 1 1 1 1	8
1 0 0 1	1 1 1 1 0 1 1	9

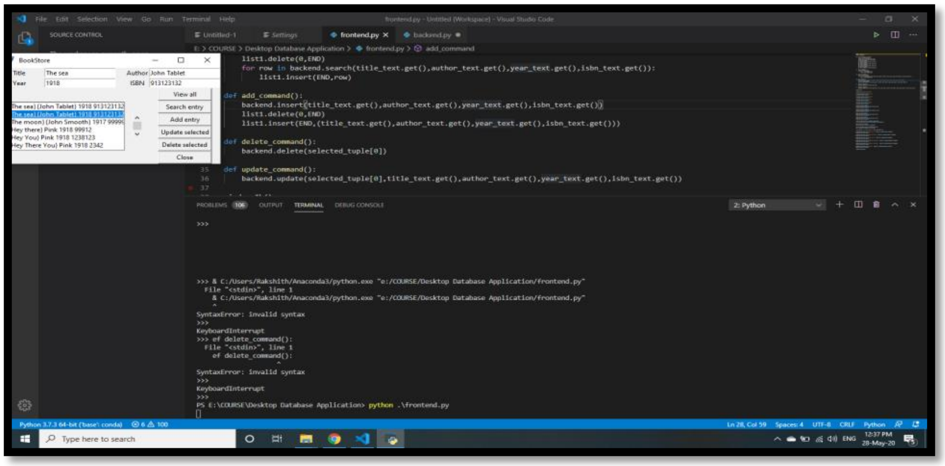
Quickgrid

DAILY ASSESSMENT FORMAT

Date:	**28/05/2020**	**Name:**	**Davis S. Patel**
Course:	**Python Course**	**USN:**	**4AL16EC045**
Topic:	**Application 5 : Build a Desktop Database Application**	**Semester & Section:**	**8th - A**
GitHub Repository:	**Davis**		

AFTERNOON SESSION DETAILS

Image of Session



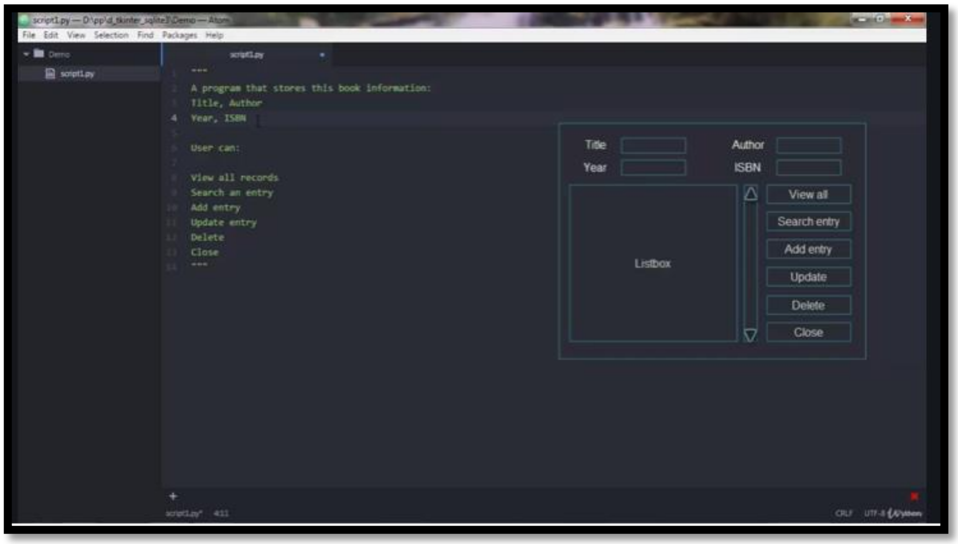
Title: The sea
Author: John Tablet
Year: 1918
ISBN: 913123132

```
The sea) (John Tablet) 1918 913123132
The sea) (John Tablet) 1918 913123132
The moon) (John Smooth) 1917 99999
Hey there) Pink 1918 99912
Hey You) Pink 1918 1238123
Hey There You) Pink 1918 2342
```

```python
    list1.delete(0,END)
    for row in backend.search(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()):
        list1.insert(END,row)

def add_command():
    backend.insert(title_text.get(),author_text.get(),year_text.get(),isbn_text.get())
    list1.delete(0,END)
    list1.insert(END,(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()))

def delete_command():
    backend.delete(selected_tuple[0])

def update_command():
    backend.update(selected_tuple[0],title_text.get(),author_text.get(),year_text.get(),isbn_text.get())
```

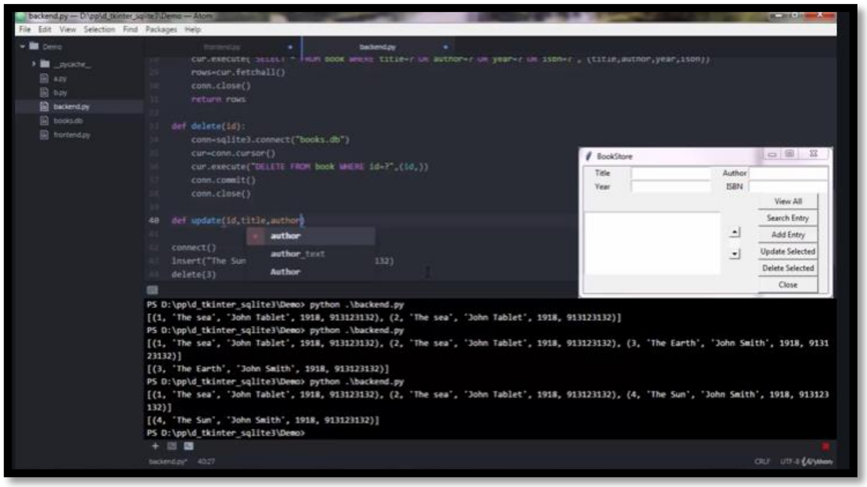
```
>>>




>>> & C:/Users/Rakshith/Anaconda3/python.exe "e:/COURSE/Desktop Database Application/frontend.py"
  File "<stdin>", line 1
    & C:/Users/Rakshith/Anaconda3/python.exe "e:/COURSE/Desktop Database Application/frontend.py"
    ^
SyntaxError: invalid syntax
>>>
KeyboardInterrupt
>>> ef delete_command():
  File "<stdin>", line 1
    ef delete_command():
                    ^
SyntaxError: invalid syntax
>>>
KeyboardInterrupt
>>>
PS E:\COURSE\Desktop Database Application> python .\frontend.py
```

Python 3.7.3 64-bit ('base': conda)
Ln 28, Col 59 Spaces: 4 UTF-8 CRLF Python
Type here to search
ENG
12:37 PM
28-May-20



script1.py — D:\pp\d_tkinter_sqlite3\Demo — Atom
File Edit View Selection Find Packages Help
Demo
script1.py
Title
Author
Year
ISBN
Listbox
View all
Search entry
Add entry
Update
Delete
Close
script1.py* 4:11
CRLF UTF-8 Python

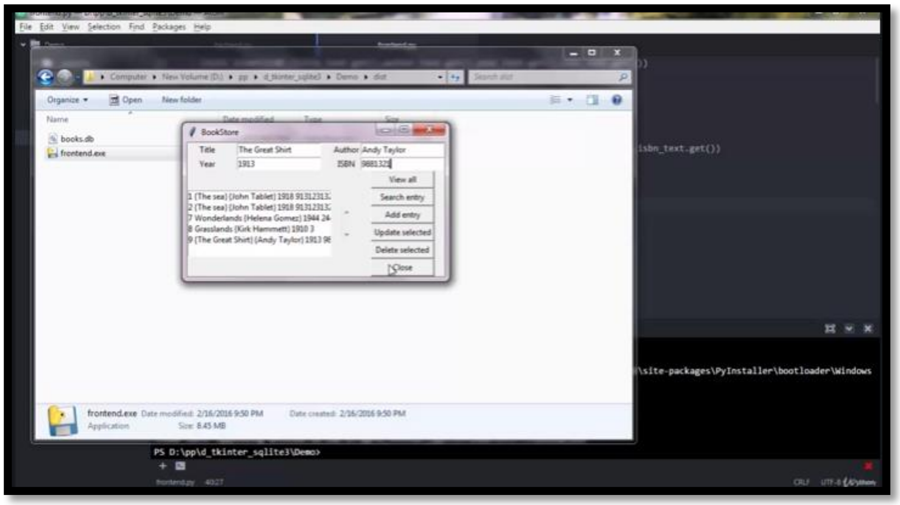
```python
"""
A program that stores this book information:
Title, Author
Year, ISBN

User can:

View all records
Search an entry
Add entry
Update entry
Delete
Close
"""
```



```
    cur.execute("SELECT * FROM book WHERE title=? OR author=? OR year=? OR isbn=?", (title,author,year,isbn))
    rows=cur.fetchall()
    conn.close()
    return rows

def delete(id):
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("DELETE FROM book WHERE id=?",(id,))
    conn.commit()
    conn.close()

def update(id,title,author)
              author
connect()     author_text
insert("The Sun                132)
delete(3)     Author
```

Title
Author
Year
ISBN

```
PS D:\pp\d_tkinter_sqlite3\Demo> python .\backend.py
[(1, 'The sea', 'John Tablet', 1918, 913123132), (2, 'The sea', 'John Tablet', 1918, 913123132)]
PS D:\pp\d_tkinter_sqlite3\Demo> python .\backend.py
[(1, 'The sea', 'John Tablet', 1918, 913123132), (2, 'The sea', 'John Tablet', 1918, 913123132), (3, 'The Earth', 'John Smith', 1918, 9131
23132)]
[(3, 'The Earth', 'John Smith', 1918, 913123132)]
PS D:\pp\d_tkinter_sqlite3\Demo> python .\backend.py
[(1, 'The sea', 'John Tablet', 1918, 913123132), (2, 'The sea', 'John Tablet', 1918, 913123132), (4, 'The Sun', 'John Smith', 1918, 913123
132)]
[(4, 'The Sun', 'John Smith', 1918, 913123132)]
PS D:\pp\d_tkinter_sqlite3\Demo>
```



Title The Great Shirt
Author Andy Taylor
Year 1913
ISBN 9881323

1 (The sea) (John Tablet) 1918 91312313
2 (The sea) (John Tablet) 1918 91312313
7 Wonderlands (Helena Gomez) 1944 24
8 Grasslands (Kirk Hammett) 1910 3
9 (The Great Shirt) (Andy Taylor) 1913 98

frontend.exe Date modified: 2/16/2016 9:50 PM Date created: 2/16/2016 9:50 PM
Application Size: 8.45 MB

```
isbn_text.get())
```

```
\site-packages\PyInstaller\bootloader\Windows
PS D:\pp\d_tkinter_sqlite3\Demo>
```

Report –

Application 5: Build a Desktop Database Application

Front-end Web Development refers to building web interfaces, specifically the parts of the website that the user will interact with. When you're browsing the web, everything you see, from images and headings to sliders and buttons is made using HTML, CSS and JavaScript, the main components to any website.

Back-end Development refers to the parts of the website that a user doesn't see or directly interact with. The back end handles application logic, algorithms, database interaction and the processing of user requests.

Frontend.py –

```
Frontend.py
from tkinter import *
import backend

def get_selected_row(event):
    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])

def view_command():
    list1.delete(0,END)
    for row in backend.view():
        list1.insert(END,row)

def search_command():
    list1.delete(0,END)
    for row in backend.search(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()):
        list1.insert(END,row)

def add_command():
    backend.insert(title_text.get(),author_text.get(),year_text.get(),isbn_text.get())
    list1.delete(0,END)
    list1.insert(END,(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()))

def delete_command():
    backend.delete(selected_tuple[0])

def update_command():

backend.update(selected_tuple[0],title_text.get(),author_text.get(),year_text.get(),isbn_text.
get())

window=Tk()
```

```
window.wm_title("BookStore")

l1=Label(window,text="Title")
l1.grid(row=0,column=0)

l2=Label(window,text="Author")
l2.grid(row=0,column=2)

l3=Label(window,text="Year")
l3.grid(row=1,column=0)

l4=Label(window,text="ISBN")
l4.grid(row=1,column=2)

title_text=StringVar()
e1=Entry(window,textvariable=title_text)
e1.grid(row=0,column=1)

author_text=StringVar()
e2=Entry(window,textvariable=author_text)
e2.grid(row=0,column=3)

year_text=StringVar()
e3=Entry(window,textvariable=year_text)
e3.grid(row=1,column=1)

isbn_text=StringVar()
e4=Entry(window,textvariable=isbn_text)
```

```
e4.grid(row=1,column=3)

list1=Listbox(window, height=6,width=35)
list1.grid(row=2,column=0,rowspan=6,columnspan=2)

sb1=Scrollbar(window)
sb1.grid(row=2,column=2,rowspan=6)

list1.configure(yscrollcommand=sb1.set)
sb1.configure(command=list1.yview)

list1.bind('<<ListboxSelect>>',get_selected_row)

b1=Button(window,text="View all", width=12,command=view_command)
b1.grid(row=2,column=3)

b2=Button(window,text="Search entry", width=12,command=search_command)
b2.grid(row=3,column=3)

b3=Button(window,text="Add entry", width=12,command=add_command)
b3.grid(row=4,column=3)

b4=Button(window,text="Update selected", width=12,command=update_command)
b4.grid(row=5,column=3)

b5=Button(window,text="Delete selected", width=12,command=delete_command)
b5.grid(row=6,column=3)
```

```
b6=Button(window,text="Close", width=12,command=window.destroy)
b6.grid(row=7,column=3)

window.mainloop()
```

Backend.py –

```
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()
    view()

def view():
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("SELECT * FROM book")
    rows=cur.fetchall()
    conn.close()
    return rows

def search(title="",author="",year="",isbn=""):
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("SELECT * FROM book WHERE title=? OR author=? OR year=? OR isbn=?",
(title,author,year,isbn))
    rows=cur.fetchall()
    conn.close()
    return rows

def delete(id):
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("DELETE FROM book WHERE id=?",(id,))
    conn.commit()
    conn.close()

def update(id,title,author,year,isbn):
    conn=sqlite3.connect("books.db")
    cur=conn.cursor()
    cur.execute("UPDATE book SET title=?, author=?, year=?, isbn=? WHERE
id=?",(title,author,year,isbn,id))
    conn.commit()
    conn.close()
```

```
connect()

insert("The Sun","John Smith",1918,913123132)

delete(3)

update(4,"The moon","John Smooth",1917,99999)

print(view())

print(search(author="John Smooth"))
```