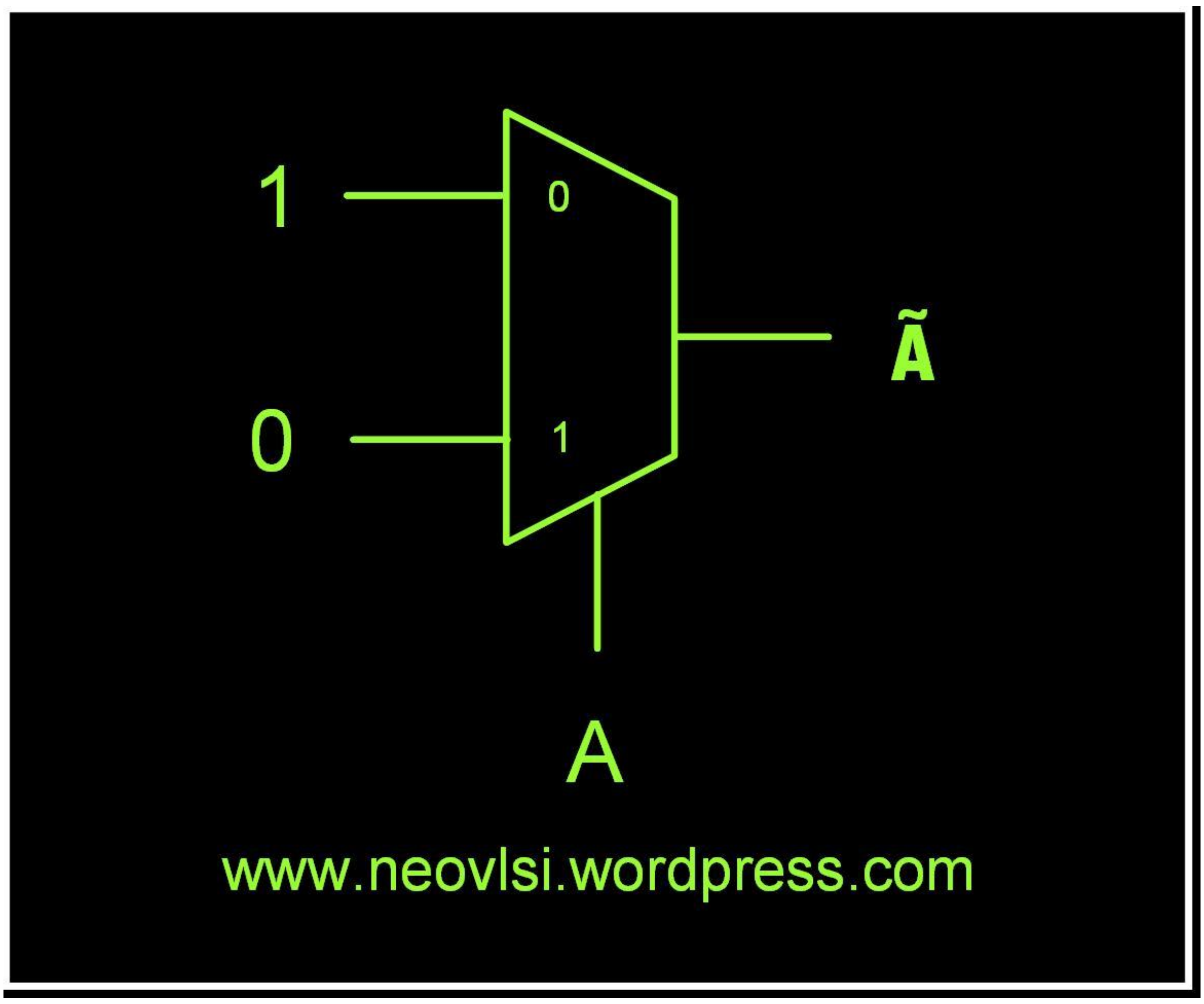
NAME	Apeksha S Shetty	USN	4AL16EC006
COURSE	Logic design	Sem and sec	8th sem A
TOPIC	Boolean equations for digital circuits, combinational circuits, design of 7 segment decoders with common anode display.		
Github repository	Apeksha-97		

IMAGE SECTION



Boolean Algebra

Is an algebra, which deals with binary numbers & binary variables. Hence, it is also called as Binary Algebra or logical Algebra. A mathematician, named George Boole had developed this algebra in 1854. The variables used in this algebra are also called as Boolean variables.

The range of voltages corresponding to Logic 'High' is represented with '1' and the range of voltages corresponding to logic 'Low' is represented with '0'.

Postulates and Basic Laws of Boolean Algebra

In this section, let us discuss about the Boolean postulates and basic laws that are used in Boolean algebra. These are useful in minimizing Boolean functions.

Boolean Postulates

Consider the binary numbers 0 and 1, Boolean variable *x* and its complement *x′*. Either the Boolean variable or complement of it is known as **literal**. The four possible **logical OR** operations among these literals and binary numbers are shown below.

$$x + 0 = x$$

$$x + 1 = 1$$

$$x + x = x$$

$$x + x' = 1$$

Similarly, the four possible **logical AND** operations among those literals and binary numbers are shown below.

$$x.1 = x$$

$$x.0 = 0$$

$$x.x = x$$

$$x.x' = 0$$

These are the simple Boolean postulates. We can verify these postulates easily, by substituting the Boolean variable with '0' or '1'.

Note− The complement of complement of any Boolean variable is equal to the variable itself. i.e., x′x′′=x.

Basic Laws of Boolean Algebra

Following are the three basic laws of Boolean Algebra.

- Commutative law
- Associative law
- Distributive law

Commutative Law

If any logical operation of two Boolean variables give the same result irrespective of the order of those two variables, then that logical operation is said to be **Commutative**. The logical OR & logical AND operations of two Boolean variables x & y are shown below

$$x + y = y + x$$

$$x.y = y.x$$

The symbol '+' indicates logical OR operation. Similarly, the symbol '.' indicates logical AND operation and it is optional to represent. Commutative law obeys for logical OR & logical AND operations.

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**. The logical OR & logical AND operations of three Boolean variables x, y & z are shown below.

$$x + y+zy+z = x+yx+y + z$$
$$x.y.zy.z = x.yx.y.z$$

Associative law obeys for logical OR & logical AND operations.

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**. The distribution of logical OR & logical AND operations of three Boolean variables x, y & z are shown below.

$$x.y+zy+z = x.y + x.z$$
$$x + y.zy.z = x+yx+y.x+zx+z$$

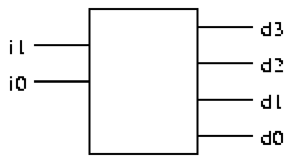
Distributive law obeys for logical OR and logical AND operations.

These are the Basic laws of Boolean algebra. We can verify these laws easily, by substituting the Boolean variables with '0' or '1'.

Conversion of MUX and Decoders to logic gates

Decoders

A decoder is a circuit which has n inputs and 2^n outputs, and outputs 1 on the wire corresponding to the binary number represented by the inputs. For example, a 2-4 decoder might be drawn like this:



and its truth table (again, really four truth tables, one for each output) is:

i_1	i_0	d_3	d_2	d_1	d_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

COMBINATIONAL CIRCUIT:

Combinational circuit is a circuit in which we combine the different gates in the circuit, for example encoder, decoder, multiplexer and demultiplexer. Some of the characteristics of combinational circuits are following −

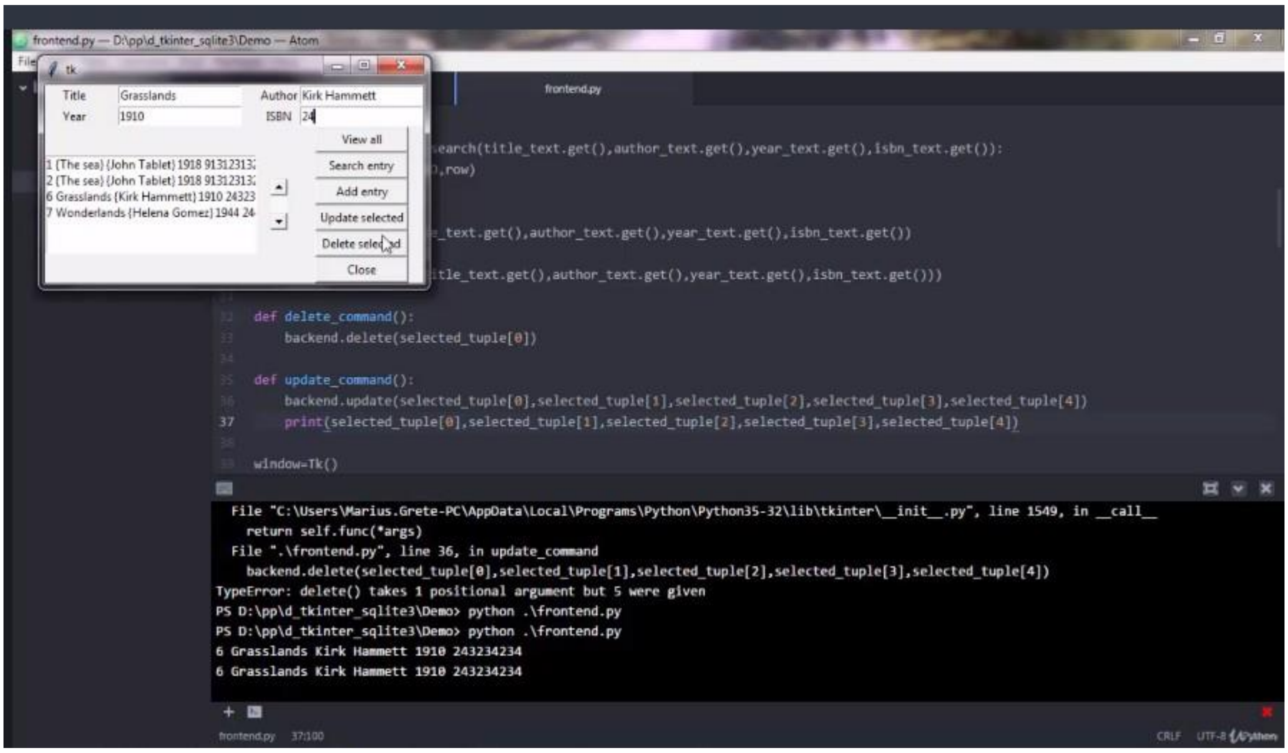
The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.

The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.

A combinational circuit can have an n number of inputs and m number of outputs

NAME	Apeksha S Shetty	USN	4AL16EC006
COURSE	Python	Sem and sec	8th sem A
TOPIC	Build a desktop database application		
Github repository	Apeksha-97		

Image section



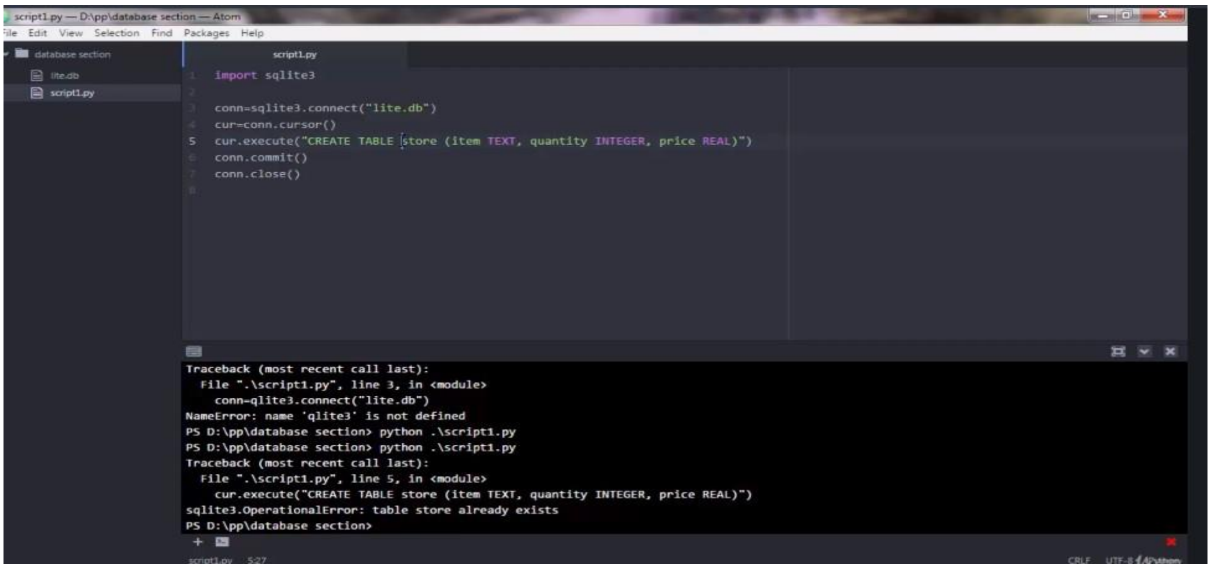
frontend.py — D:\pp\d_tkinter_sqlite3\Demo — Atom

Title	Grasslands	Author	Kirk Hammett
Year	1910	ISBN	24

```
1 {The sea} {John Tablet} 1918 91312313:
2 {The sea} {John Tablet} 1918 91312313:
6 Grasslands {Kirk Hammett} 1910 24323
7 Wonderlands {Helena Gomez} 1944 24
```

```
search(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()):
),row)

_text.get(),author_text.get(),year_text.get(),isbn_text.get())

itle_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],selected_tuple[1],selected_tuple[2],selected_tuple[3],selected_tuple[4])
    print(selected_tuple[0],selected_tuple[1],selected_tuple[2],selected_tuple[3],selected_tuple[4])

window=Tk()
```

```
  File "C:\Users\Marius.Grete-PC\AppData\Local\Programs\Python\Python35-32\lib\tkinter\__init__.py", line 1549, in __call__
    return self.func(*args)
  File ".\frontend.py", line 36, in update_command
    backend.delete(selected_tuple[0],selected_tuple[1],selected_tuple[2],selected_tuple[3],selected_tuple[4])
TypeError: delete() takes 1 positional argument but 5 were given
PS D:\pp\d_tkinter_sqlite3\Demo> python .\frontend.py
PS D:\pp\d_tkinter_sqlite3\Demo> python .\frontend.py
6 Grasslands Kirk Hammett 1910 243234234
6 Grasslands Kirk Hammett 1910 243234234
```



script1.py — D:\pp\database section — Atom

```
import sqlite3

conn=sqlite3.connect("lite.db")
cur=conn.cursor()
cur.execute("CREATE TABLE store (item TEXT, quantity INTEGER, price REAL)")
conn.commit()
conn.close()
```

```
Traceback (most recent call last):
  File ".\script1.py", line 3, in <module>
    conn=qlite3.connect("lite.db")
NameError: name 'qlite3' is not defined
PS D:\pp\database section> python .\script1.py
PS D:\pp\database section> python .\script1.py
Traceback (most recent call last):
  File ".\script1.py", line 5, in <module>
    cur.execute("CREATE TABLE store (item TEXT, quantity INTEGER, price REAL)")
sqlite3.OperationalError: table store already exists
PS D:\pp\database section>
```

REPORT:

CONNECT BACKEND TO FRONTEND

Python has been the most trending programming language used for object oriented programming. With python you can run simple statement over and over again without having to compile a whole program of which it's output functionality is superb.

There are exactly two ways to perform what you are looking for, i.e., to insert data from data services of your backend into your frontend layout as content.

1. Server Side Rendering- This is how most and traditional websites specifically those are based on some CMS technology uses. The idea is to have some HTML template files with some placeholder for content that are later pushed by the backend program before sending to client browsers. In Python, you can have Jinja template framework at your disposal. You can wrap up variables into double braces and later providing values for each variable and compile them to static HTML file to render them on browser.

2.Hybrid- You may also however combine these two technique by using some server side rendering with a JavaScript plugin library with Ajax support such as jQuery.