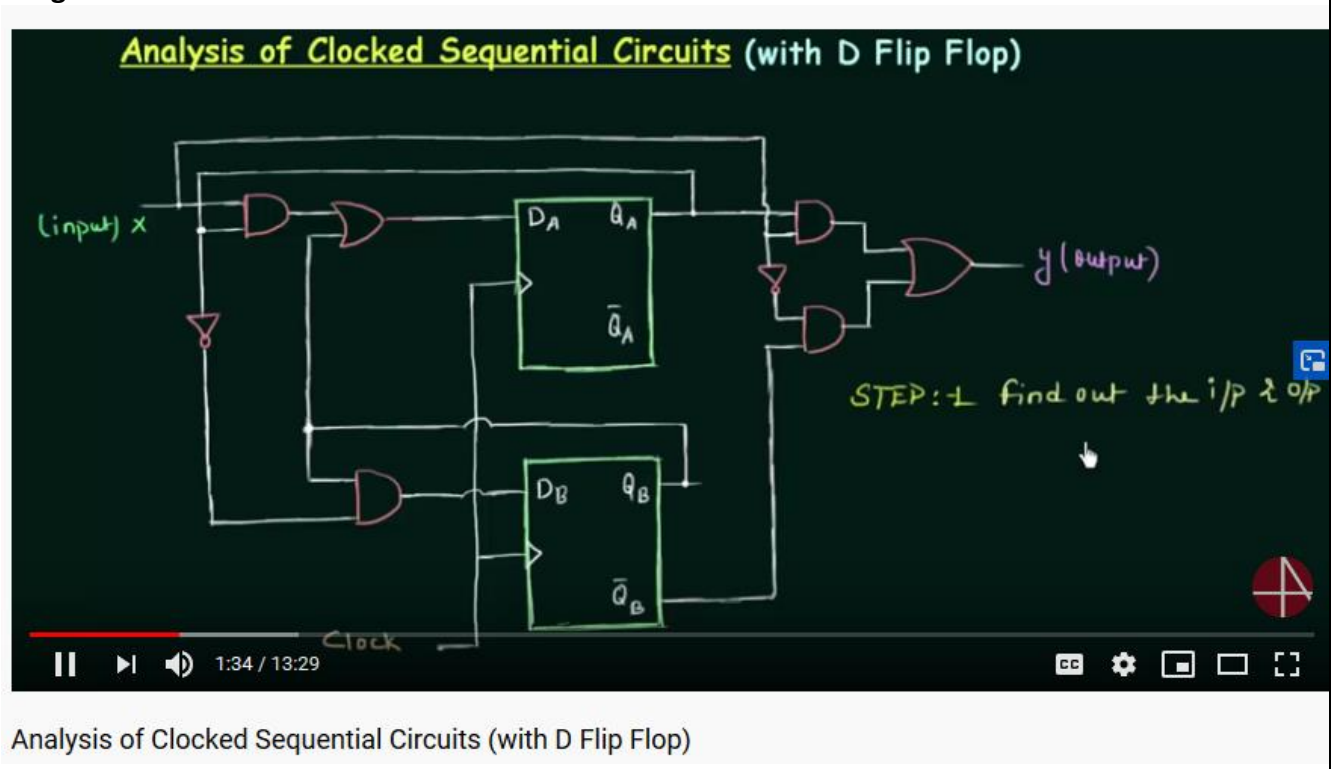
DAILY ASSESSMENT FORMAT

Date:	**29/05/2020**	**Name:**	**Nishanth**
Course:	**Logic Design**	**USN:**	**4al17ec063**
Topic:	**1.Analysis of clocked sequential circuits** **2. Digital clock design**	**Semester & Section:**	**6thb-section**
GitHub Repository:	**nishanthvr**		

FORENOON SESSION DETAILS

Image of session



Analysis of Clocked Sequential Circuits (with D Flip Flop)

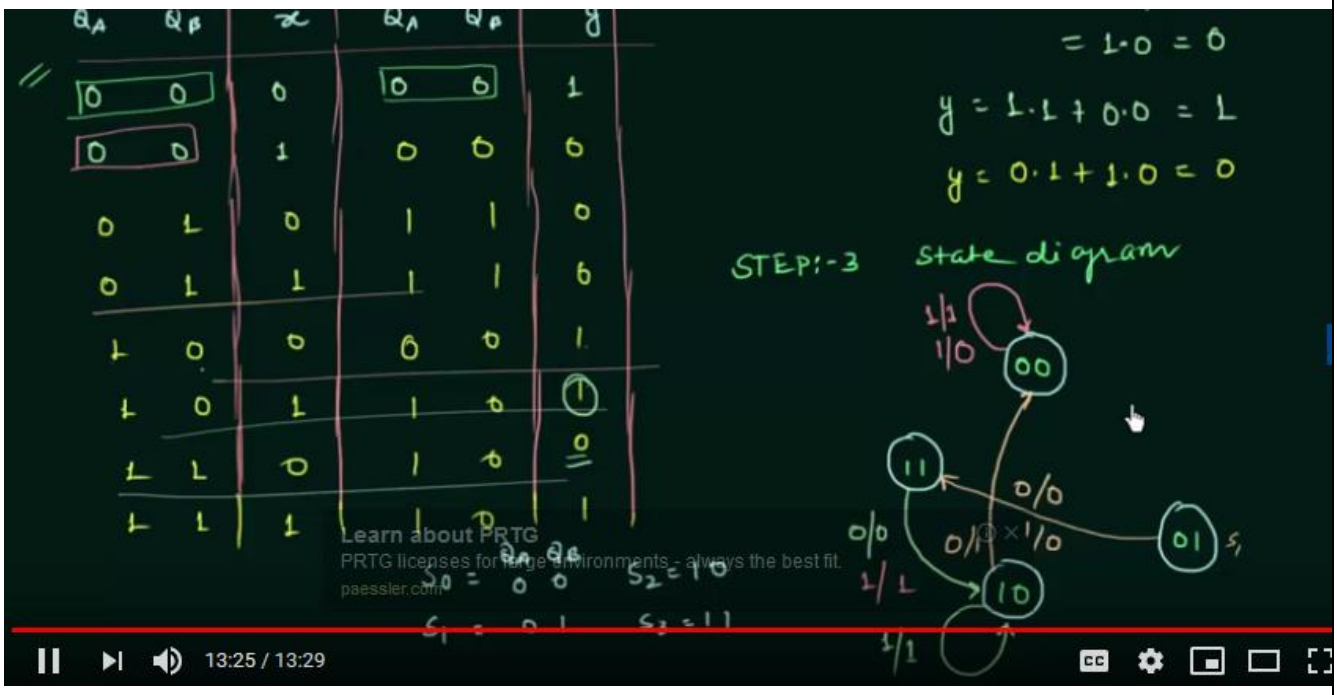
Combinational Circuits :

Are circuits made up of different types of logic gates. A logic gate is a basic building block of any electronic circuit. The output of the combinational circuit depends on the values at the input at any given time. The circuits do not make use of any memory or storage device

The D flip-flop tracks the input, making transitions with match those of the input D. The D stands for "data"; this flip-flop stores the value that is on the data line. It can be thought of as a basic memory cell. A D flip-flop can be made from a set/reset flip-flop by tying the set to the reset through an inverter

D flip flop truth table

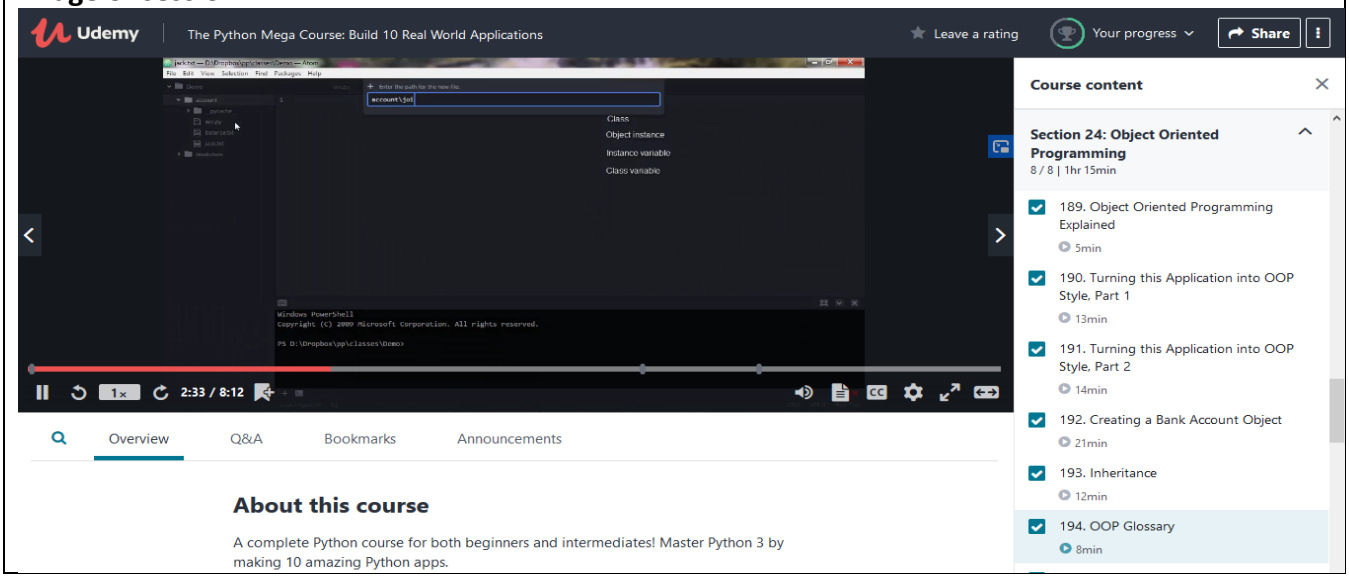
The term digital in electronics represents the data generation, processing or storing in the form of two states. The two states can be represented as HIGH or LOW, positive or non-positive, set or reset which is ultimately binary



Date:	**29/05/2020**	**Name:**	**Nishanth**
Course:	**Python**	**USN:**	**4al17ec063**
	Object Oriented Programming	**Semester & Section:**	**6th and b section**

AFTERNOON SESSION DETAILS

Image of session



Here are the *frontend.py* and *backend.py* scripts in OOP style. To execute this program you should execute the *frontend.py* file.

```
#frontend.py

1. from tkinter import *
2. from backend import Database
3.
4. database=Database("books.db")
5.
6. class Window(object):
7.
8.     def __init__(self,window):
9.
10.         self.window = window
11.
12.         self.window.wm_title("BookStore")
13.
14.         l1=Label(window,text="Title")
15.         l1.grid(row=0,column=0)
16.
17.         l2=Label(window,text="Author")
18.         l2.grid(row=0,column=2)
19.
20.         l3=Label(window,text="Year")
21.         l3.grid(row=1,column=0)
22.
23.         l4=Label(window,text="ISBN")
24.         l4.grid(row=1,column=2)
25.
26.         self.title_text=StringVar()
27.         self.e1=Entry(window,textvariable=self.title_text)
28.         self.e1.grid(row=0,column=1)
29.
30.         self.author_text=StringVar()
31.         self.e2=Entry(window,textvariable=self.author_text)
32.         self.e2.grid(row=0,column=3)
33.
34.         self.year_text=StringVar()
35.         self.e3=Entry(window,textvariable=self.year_text)
36.         self.e3.grid(row=1,column=1)
37.
38.         self.isbn_text=StringVar()
39.         self.e4=Entry(window,textvariable=self.isbn_text)
40.         self.e4.grid(row=1,column=3)
41.
42.         self.list1=Listbox(window, height=6,width=35)
43.         self.list1.grid(row=2,column=0,rowspan=6,columnspan=2)
44.
45.         sb1=Scrollbar(window)
46.         sb1.grid(row=2,column=2,rowspan=6)
47.
48.         self.list1.configure(yscrollcommand=sb1.set)
49.         sb1.configure(command=self.list1.yview)
50.
51.         self.list1.bind('<<ListboxSelect>>',self.get_selected_row)
52.
53.         b1=Button(window,text="View all",
   width=12,command=self.view_command)
```

```
54.          b1.grid(row=2,column=3)
55.
56.          b2=Button(window,text="Search entry",
   width=12,command=self.search_command)
57.          b2.grid(row=3,column=3)
58.
59.          b3=Button(window,text="Add entry",
   width=12,command=self.add_command)
60.          b3.grid(row=4,column=3)
61.
62.          b4=Button(window,text="Update selected",
   width=12,command=self.update_command)
63.          b4.grid(row=5,column=3)
64.
65.          b5=Button(window,text="Delete selected",
   width=12,command=self.delete_command)
66.          b5.grid(row=6,column=3)
67.
68.          b6=Button(window,text="Close", width=12,command=window.destroy)
69.          b6.grid(row=7,column=3)
70.
71.      def get_selected_row(self,event):
72.          index=self.list1.curselection()[0]
73.          self.selected_tuple=self.list1.get(index)
74.          self.e1.delete(0,END)
75.          self.e1.insert(END,self.selected_tuple[1])
76.          self.e2.delete(0,END)
77.          self.e2.insert(END,self.selected_tuple[2])
78.          self.e3.delete(0,END)
79.          self.e3.insert(END,self.selected_tuple[3])
80.          self.e4.delete(0,END)
81.          self.e4.insert(END,self.selected_tuple[4])
82.
83.      def view_command(self):
84.          self.list1.delete(0,END)
85.          for row in database.view():
86.              self.list1.insert(END,row)
87.
88.      def search_command(self):
89.          self.list1.delete(0,END)
90.          for row in
   database.search(self.title_text.get(),self.author_text.get(),self.year_text.g
   et(),self.isbn_text.get()):
91.              self.list1.insert(END,row)
92.
93.      def add_command(self):
94.
   database.insert(self.title_text.get(),self.author_text.get(),self.year_text.g
   et(),self.isbn_text.get())
95.          self.list1.delete(0,END)
96.
   self.list1.insert(END,(self.title_text.get(),self.author_text.get(),self.year
   _text.get(),self.isbn_text.get()))
97.
98.      def delete_command(self):
99.          database.delete(self.selected_tuple[0])
100.
101.     def update_command(self):
```

```
102.
   database.update(self.selected_tuple[0],self.title_text.get(),self.author_text
   .get(),self.year_text.get(),self.isbn_text.get())
103.

window=Tk()

104. Window(window)
105. window.mainloop()

   #backend.py

106. import sqlite3
107. class Database:
108.     def __init__(self, db):
109.         self.conn=sqlite3.connect(db)
110.         self.cur=self.conn.cursor()
111.         self.cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER
   PRIMARY KEY, title text, author text, year integer, isbn integer)")
112.         self.conn.commit()
113.     def insert(self,title,author,year,isbn):
114.         self.cur.execute("INSERT INTO book VALUES
   (NULL,?,?,?,?)",(title,author,year,isbn))
115.         self.conn.commit()
116.     def view(self):
117.         self.cur.execute("SELECT * FROM book")
118.         rows=self.cur.fetchall()
119.         return rows
120.     def search(self,title="",author="",year="",isbn=""):
121.         self.cur.execute("SELECT * FROM book WHERE title=? OR author=? OR
   year=? OR isbn=?", (title,author,year,isbn))
122.         rows=self.cur.fetchall()
123.         return rows
124.     def delete(self,id):
125.         self.cur.execute("DELETE FROM book WHERE id=?",(id,))
126.         self.conn.commit()
127.     def update(self,id,title,author,year,isbn):
128.         self.cur.execute("UPDATE book SET title=?, author=?, year=?, isbn=?
   WHERE id=?",(title,author,year,isbn,id))
129.         self.conn.commit()
130.     def __del__(self):
131.         self.conn.close()
```